콤퓨터조직과 구성방식

김책공업종합대학출판사
주체91

콤퓨터조직과
구성방식

김책공업종합대학출판사

차 례

제 5 장. 외부기억기

제 6 장. 입출력

제 7 장. 조작체계의 지원

제 3 편. 중앙처리장치

제 8 장. 콤퓨터산수연산

제 9 장. 명령모임: 특성과 기능

제 10 장. 명령모임: 주소화방식과 형식

제 11 장. CPU 구조와 기능

제 12 장. 축소명령모임콤퓨터

제 13 장. 명령준위병렬화와 슈퍼스칼라처리장치

제 4 편. 조종장치

제 14 장. 조종장치조작

제 15 장. 마이크로프로그람조종

제 5 편. 병렬조직

제 16 장. 병렬처리

머 리 말

목 적

이 책에서는 콤퓨터의 구조와 기능에 대하여 서술하였다. 목적은 현대콤퓨터체계들의 본질과 특성을 될수록 명백하게 그리고 충분히 주자는데 있다.

이러한 과제가 제기되게 된것은 다음과 같은 몇가지 원인과 관련된다. 첫째로, 가격이 수딸라정도인 한소편극소형처리소자들로부터 수천만딸라정도인 슈퍼콤퓨터에 이르기까지 《콤퓨터》라고 말할수 있는 제품들이 아주 다양한것이다. 이러한 다양성은 가격뿐아니라 크기와 성능, 응용의 견지에서도 찾아 볼수 있다. 둘째로, 콤퓨터의 속도가 끊임없이 계속 높아 지고 있는것이다. 속도가 이렇게 높아 지는것은 콤퓨터요소를 만드는 집적회로기술로부터 그 요소들을 결합한 병렬조직개념의 리용범위를 확장하는데 이르기까지의 전반적인 콤퓨터기술이 발전한것과 관련된다.

콤퓨터가 이렇게 다양하고 속도도 서로 다르지만 콤퓨터분야에서는 명백한 기초개념들이 시종일관하게 철저히 응용되고 있다. 이 개념들이 어떻게 응용되는가는 현재의 기술상태와 설계자가 설정한 가격/성능목표에 관계된다. 이 책의 목적은 콤퓨터조직과 구성방식의 기초를 고찰하고 그것을 오늘날의 콤퓨터설계지표들과 련관시키는것이다.

소제목들에는 이 책에서 취급한 문제들과 고찰방법들이 제시되여 있다. 성능이 높은 콤퓨터체계를 설계하는것은 어느 때나 중요하였지만 지금보다는 이 요구가 결코 높지 않았으며 더 어렵지도 않았다. 오늘에 와서는 처리장치의 속도, 기억기속도, 기억기용량, 자료의 호상결합속도를 비롯하여 콤퓨터체계의 모든 기본성능들이 빨리 개선되고 있다. 그리고 이 기본성능들은 발전속도가 각이하다. 이것은 모든 요소들의 성능과 그 리용의 견지에서 최대인 체계를 실현하는데서 어려운 문제로 제기되고 있다. 그리하여 콤퓨터설계에서는 한 부분에서 성능이 제대로 보장되지 못하는것을 다른 부분에서의 구조와 기능을 변화시키는 방법으로 보상하려는 시도가 더욱더 늘어 나고 있다. 이 책을 통하여 여러가지 설계방법들을 적용한 수법들을 알수 있다.

콤퓨터체계는 다른 체계들과 마찬가지로 련관되여 있는 요소들의 묶음으로 구성된다. 이 체계는 구성부분들을 서로 련결하는 방법인 구조라는 말과 개별적인 구성부분들의 조작인 기능이라는 말로 특징 지어 진다. 그리고 콤퓨터조직은 계층적으로 이루어 져 있다. 매개 구성부분들은 더 작은 구성부분으로 분해하고 그것들의 구조와 기능을 설명하는 방법으로 서술할수 있다. 명백하면서도 쉽게 리해하도록 하기 위하여 이 책에서는 계층구성을 다음과 같이 웃준위로부터 아래준위로 내려가면서 서술하였다.

- **콤퓨터체계:** 기본구성부분들은 처리장치, 기억기, I/O 장치이다.
- **처리장치:** 기본구성부분들은 조종장치, 등록기, ALU, 명령실행장치이다.
- **조종장치:** 기본구성은 조종기억기, 마이크로명령순서조종론리, 등록기이다.

목적은 이 내용들을 서술함에 있어서 새 내용이 명백한 전후관계속에서 서술되도록 하는데 있다. 이것은 독자들이 될수록 향방을 잃지 않고 아래로부터 우로 올라 가는 식으로 학습하는것보다 의욕을 보다 높여 주기 위한것이다.

이 책의 전반에서 체계의 매개 측면들은 구성방식(체계의 이 측면은 기계어프로그람작성자들에게 직관적인것이다.)의 관점과 조직(구성방식을 련상할수 있게 하는 조작장치들과 그것들의 결합)의 관점에서 고찰되고 있다.

체계들의 실례

책에서 취급한 여러 종류의 콤퓨터들에 대한 실례들은 현재 쓰고 있는 개념들을 명백히 하거나 보충하는데 리용되였다. 대부분의 실례들은 두개의 콤퓨터계렬 즉 인텔의 Pentium Ⅱ와 PowerPC로부터 취하였다(Pentium Ⅲ은 다매체명령들이 확장되였을뿐 본질적으로는 Pentium Ⅱ와 같다.).

이 두 체계들에는 콤퓨터설계의 현재추세들이 대부분 반영되여 있다. Pentium Ⅱ는 본질적으로 RISC 중심의 복합명령모임콤퓨터(CISC)이지만 PowerPC는 축소명령모임콤퓨터(RISC)이다. 이 두 체계들은 슈퍼스칼라설계원리들을 리용하였으며 둘다 다중처리장치의 구성방식을 지원하고 있다.

본문의 내용구성

이 책은 5개의 편으로 구성되였다.

제1편-일반개념: 이 편에서는 이 책의 뒤부분들에 대한 예비적인 고찰과 전후관계를 주고 있다.

제2편-콤퓨터체계: 콤퓨터체계는 처리장치, 기억기, I/O장치들과 이 기본구성부분들의 결합들로 이루어 져 있다. 이 편에서는 제3편에서 고찰하는 아주 복잡한 처리장치를 제외하고 나머지부분체계들도 고찰하였다.

제3편-중앙처리장치: CPU는 조종장치, 등록기, 산수-론리연산장치, 명령실행장치와 이 구성부분들의 결합으로 이루어 져 있다. 이 편에서는 명령모임설계와 자료형식들과 같은 방식적인 지표들을 고찰한다. 또한 관흐름조종과 같은 구성지표들도 고찰한다.

제4편-조종장치: 조종장치는 처리장치의 여러 구성부분들을 동작시키는 처리장치의 한 부분이다. 이 편에서는 조종장치의 기능과 마이크로프로그람조종방식을 리용하는 조종부의 실현방법을 고찰한다.

제5편-병렬처리구성: 마지막부분에서는 다중처리장치와 벡토르처리구성에 포함되는 일부 지표들을 고찰한다.

장들에 대한 상세한 개괄은 제1장의 뒤에서 준다.

교원들과 대학생들을 위한 인터네트봉사

이 책에는 교원들과 학생들에게 도움을 주는 Web싸이트도 소개되여 있다. 이 싸이트는 다른 련관 있는 싸이트들과의 련결, PDF(Adobe Acrobat)형식으로 이 책에 있는 모든 그림들의 뚜렷한 원본들, 책의 인터네트전자우편목록과 관련된 서명된 정보를 포함하고 있다. 이 Web페지는 http://www. Shore .net/~ws/COA5e html이다. 인터네트전자우편목록은 이 책을 리용하는 교원들이 호상간 그리고 이 책의 저자와 정보, 제안, 질문들을 서로 교환할수 있게 설치되였다. 편집상 혹은 인쇄상 오유들을 발견하는 즉시로 http://www. Shore .net/~ws를 찾으면 이 책의 오유목록을 찾아 대조할수 있다.

많은 교원들에게 있어서 콤퓨터조직과 구성방식과정안의 중요한 부분은 대학생들이 본문으로부터 개념들을 보다 확고히 알도록 제손으로 체험하게 하는 방식으로 된 교수과제 또는 교수과제묶음을 설정하는것이다. 이 책은 과정안집행에서 교수과제의 요소들을 포함시키도록 하는데 비할바 없는 도움을 준다. 교원들의 지도안에는 교수과제를 배당하고 구성하기 위한 지침과 함께 본문의 여러가지 문제를 포괄하는 제안된 교수과제묶음이 포함되여야 한다.

- **연구과제:** 지도안에는 대학생들이 Web 혹은 문헌에서 개별적인 제목들을 연구하고 보고서를 작성하도록 가르치는 연구과제들을 포함한다.
- **모의과제:** 지도안에는 모의프로그람인 SimpleScalar의 리용과 관련된 지원을 포함한다. SimpleScalar는 콤퓨터조직과 구성방식설계를 위한 항목들을 탐색하는데 리용된다.
- **읽기/보고서과제:** 지도안에는 매 장별로 된 참고문헌에 있는 론문들의 목록을 포함시키고 대학생별로 읽도록 과제를 주고 그에 따라 짧은 보고서를 작성하도록 할 수 있다.

자세한것은 부록 II를 참고하면 된다.

이 책에서 새로운것은 무엇인가

이 책의 제 4 판이 출판된 때로부터 4 년동안에 이 분야에서는 련속적인 혁신과 발전이 이룩되였다. 이 책의 새 판에서는 전반적으로는 포괄적인 적용범위를 그대로 유지하면서 새로운 변화들을 더 반영하려고 시도하였다. 이 책의 수정보충을 위하여 이 과목을 배워 주는 많은 교수들이 이 책의 제 4 판을 전반적으로 후열하였다. 결과 여러 개소에서 론의들이 명백해 지고 확고해 졌으며 실례들이 개선되였다. 또한 새로운 《부분별 시험》 문제들이 보충되였다.

교육자들과 사용자들에 대한 친절성을 도모하기 위하여 노력한 결과 이 책전반이 본질적으로 달라 졌다. 장별 구성은 대체적으로 같이 하였지만 수많은 자료들이 수정되고 새로운 자료들이 첨부되였다. 가장 중요한 변화는 다음과 같다.

- **빛기억기:** 빛기억기에 대한 자료에는 자기광학식빛기억장치들을 더 포함시켰다.
- **슈퍼스칼라설계:** 슈퍼스칼라설계에 대한 장에는 더 상세한 론의와 두개의 새로운 모형 즉 UltraSpare II와 MIPS R10000을 포함시켰다.
- **다매체명령모임:** Pentium II와 Pentium III에서 리용되는 MMX명령모임이 소개되였다.
- **예견적인 실행과 리론상의 과제:** 이 판에서는 Intel과 Hewlett-Packard로부터 새로운 IA-64 구성방식의 설계에서 중심인 새로운 개념들에 대하여 론의하였다.
- **SMPS, clusters, NUMA 체계:** 병렬조직에 대한 장은 완전히 수정되였다. 그것은 대칭다중처리장치체계들(SMPs), 집합콤퓨터, 비균일기억기호출체계(NUMA)들에 대한 비교와 상세한 서술들을 새롭게 포함하였다.
- **확장된 지도서지원:** 이미 언급하였지만 이 책에서는 새로운 계획을 위한 확장된 지원을 제공하였다. 이 책에서 제공된 Web싸이트지원 역시 확장된것이다.

제 1 편. 일반개념

제 1 편의 중심

제 1 편에서는 이 책의 뒤부분에 대한 지식과 내용을 준다. 다시말하여 콤퓨터조직과 구성방식의 기본개념을 준다.

제 1 편의 장별 내용

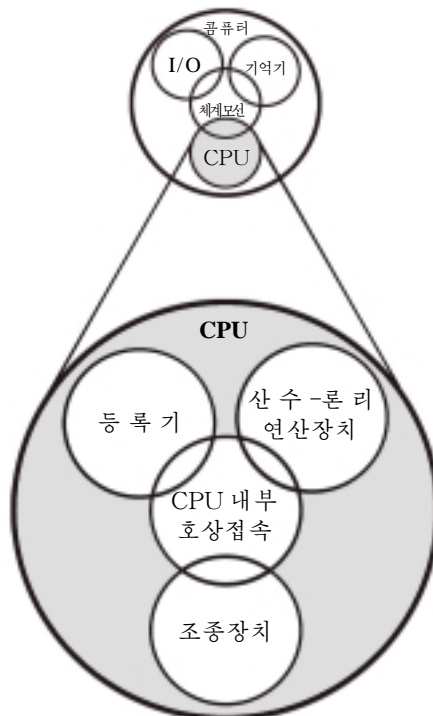
제 1 장. 안내

제 1 장에서는 계층체계로서 콤퓨터의 개념을 준다. 콤퓨터는 부분적요소들의 구조로 고찰할수 있으며 그의 기능은 서로 결합하고 있는 구성요소들의 집체적인 기능으로 표현된다. 매개 구성요소들은 내부구조로 표시된다. 이 장에서는 이 계층적고찰의 기본준위들을 소개하였다. 책의 뒤부분은 우에서부터 아래까지의 이 준위들로 편성되였다.

제 2 장. 콤퓨터의 발전과 성능

제 2 장에서는 기계적장치들로부터 현재체계까지의 콤퓨터들의 간단한 발전력사를 주었다. 이 력사에서는 일부 중요한 콤퓨터설계특성을 강조하며 콤퓨터구조의 높은 준위를 고찰한다. 이 장에서는 기본주제로서 성능에 대한 설계를 고찰한다. 그리고 성능특성이 크게 차이나는 구성부분을 리용하는 경우 균형을 맞추는것의 중요성을 강조한다.

제 1 장. 소개

◆◆

◆ 콤퓨터체계의 기본구성요소들은 CPU, 주기억기, I/O 보조체계이며 일부 이 모든 구성부분들의 호상접속이다. 그리고 CPU 는 조종장치, 산수-론리연산장치, 내부등록기 그리고 이것들의 호상간 접속기구들로 구성된다.

◆◆

제 1 절. 조직과 구성방식

콤퓨터체계서술에서는 **콤퓨터구성방식**과 **콤퓨터조직**사이의 차이를 똑똑히 갈라 놓아야 한다. 이 용어들의 정확한 정의를 주기는 어렵지만 각기 포괄하는 일반적령역에 대하여서는 공통된 견해가 있다([VRAN80], [SIEW82], [BELL78a] 참고) .

콤퓨터구성방식이란 프로그람작성자의 견지에서 본 체계의 속성 혹은 프로그람의 론리적실행에 직접적영향을 미치는 속성을 말한다. 콤퓨터조직이란 구성방식지표들을 실현하는 연산장치들과 그것들의 호상접속을 말한다. 구성방식적속성들의 실례는 명령모임, 여러가지 형식(실례로 수자, 문자)의 자료들을 표현하는데 리용되는 비트의 길이, I/O 장치 그리고 기억기주소화기술들이다. 구성적속성들은 프로그람작성자에게 명백히 구별되는 하드웨어세부장치들 즉 조종신호, 콤퓨터와 주변장치사이의 대면부 그리고 리용되는 기억기기술과 같은것들이다.

실례로 콤퓨터가 곱하기명령을 가질것을 방식적설계계획으로 한다고 하자. 이때 전용곱하기장치로 혹은 체계의 더하기장치를 반복적으로 리용하는 기구로 실현하는 구성계획을 세울수 있다. 이것을 어떻게 구성하겠는가 하는 결심채택은 곱하기명령리용에 예견된 주파수, 두 방법의 상대적속도, 가격, 전용곱하기장치의 물리적크기에 기초한다.

력사적으로 그리고 오늘까지 구성방식과 조직사이의 구별은 중요한 문제로 되고 있다. 많은 콤퓨터제작자들은 모두가 구성방식은 같지만 조직이 서로 다른 여러가지 계렬의 콤퓨터형들을 내놓고 있다. 그렇기때문에 계렬내의 서로 다른 형들은 서로 다른 가격과 성능특성들을 가진다. 그러므로 구성방식은 오래동안 유지되고 있지만 그 조직은 기술변화에 따라 변화된다. 이 두 현상의 적절한 실례로는 IBM System/370 구성방식이다. 이 방식은 1970 년에 처음으로 소개되여 형번호에 이 년도를 포함시켰다. 구매자들은 요구에 따라 눅고 속도가 뜬 형을 살수 있었으며 만일 요구가 더 높아 지면 후에 이미 개발된 프로그람을 버리지 않고도 더 비싼 비용으로 더 빠른 형으로 갱신할수 있었다. 그후 IBM 은 낡은 기술을 교체하여 개선된 기술로 구매자들에게 보다 속도가 높고 가격이 눅은 새로운 형을 소개하였다. 이 새로운 형의 콤퓨터는 같은 구성방식을 유지함으로써 구매자들이 쏘프트웨어를 구입하는데 비용을 새로 지출하지 않게 하였다. 주목되는것은 SYSTEM/370 구성방식이 약간씩 강화되면서도 오늘까지도 IBM 의 대형콤퓨터생산흐름선의 구성방식을 그대로 유지하고 있는것이다.

극소형콤퓨터체계에서 구성방식과 조직사이에는 대단히 밀접한 관계가 있다. 기술의 변화는 조직에만 영향을 미치는것이 아니라 더 위력하고 더 풍부한 구성방식들이 나오게 하는데도 영향을 미치였다. 일반적으로 이 소형콤퓨터들에 있어서는 세대에 따르는 호환성의 요구가 적다. 그러므로 조직과 구성방식사이에는 설계결심채택에서 보다 호상영향이 크다. 흥미 있는 실례는 제 12 장에서 설명하는 축소명령모임콤퓨터(RISC)이다.

이 책에서는 콤퓨터조직과 콤퓨터구성방식을 설명하였다. 중점은 구성측면에 두었다.

그러나 콤퓨터조직이 특정한 구성방식지표의 실현을 위해서 설계되는것만큼 콤퓨터조직을 철저히 고찰하는것은 역시 구성방식에 대한 자세한 고찰도 동반할것을 요구한다.

제 2 절. 구조와 기능

콤퓨터는 복합체계이다. 즉 현 시대 콤퓨터들은 요소적인 수백만개의 전자부분품들을 포함한다. 그러면 어떻게 그것들을 명백히 서술할수 있겠는가. 기본문제는 콤퓨터를 포함하는 대부분 복합체계의 계층적본질을 식별하는것이다[SIMO69]. 계층체계는 서로 련관된 보조체계들의 묶음이다.

복합체계의 계층성은 그것들에 대한 설계와 서술에 있어서 가장 중요하다. 설계자는 한번에 체계를 특수한 준위들로 분할할것을 요구한다. 매 준위에서 체계는 구성부분들과 그것들의 호상결합들로 이루어 진다. 매 준위에서의 동작은 다음의 낮은 준위의 단순하고 추상적인 성질에 의존한다. 설계자는 매 준위에서 구조와 기능에 관심을 돌린다.

- **구조**: 구성부분들을 호상 련관시키는 방법
- **기능**: 구성부분으로서의 개별적인 구성요소의 연산기능

계층을 서술할 때에는 제일 아래준위로부터 시작하여 체계를 구축하는 방법과 웃준위로부터 고찰하여 보조적인 부분들로 체계를 분해하는 두가지 방법을 선택한다. 여러가지 론의로부터 우에서부터 아래로의 방법이 가장 명백하고 효과적이라는것을 알수 있다[WEIN75].

이 책에서 취급한 방법은 이 고찰에 따른다. 콤퓨터체계는 우에서부터 아래로의 방향으로 서술된다. 이 책에서는 기본구성부분들로부터 시작하여 구조와 기능을 제일 낮은 계층까지 서술하였다. 이 절의 뒤부분에서는 이 방법을 아주 간단히 고찰하였다.

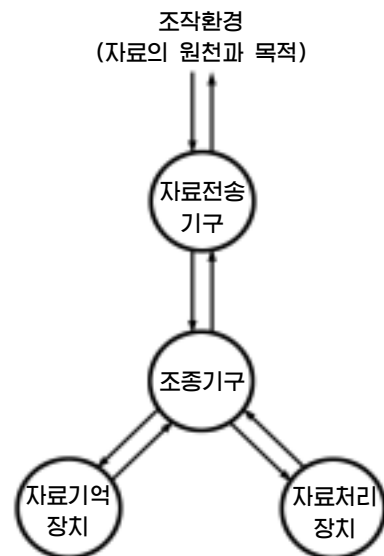
1. 기능

콤퓨터의 구조와 기능화는 실제로 간단하다. 그림 1-1 에서는 콤퓨터가 실행할수 있는 기본기능을 보여 주었다. 일반적인 용어로는 다음의 4가지가 있다.

- 자료처리
- 자료기억
- 자료전송
- 조종

그림 1-1. 콤퓨터의 기능고찰

물론 콤퓨터는 자료를 처리할수 있어야 한다. 자료는 형식이 매우 다양하며 처리요구의 범위가 넓다. 그러나 자료처리에는 몇가지 기초적방법 혹은 형식이 있을뿐이다.

또한 콤퓨터는 반드시 자료를 기억하여야 한다. 만일 콤퓨터가 가동상태에서 자료를 처리하고 있다면 콤퓨터는 임의의 주어 진 순간에 처리되고 있는 중간결과들을 림시적으로 기억하여야 한다. 따라서 림시적인 자료기억기능이 있어

야 한다. 마찬가지로 중요한것은 콤퓨터가 오랜 기간 자료기억기능을 수행하여야 한다는 것이다. 자료파일들은 후에 수정하고 갱신하기 위하여 콤퓨터에 기억된다.

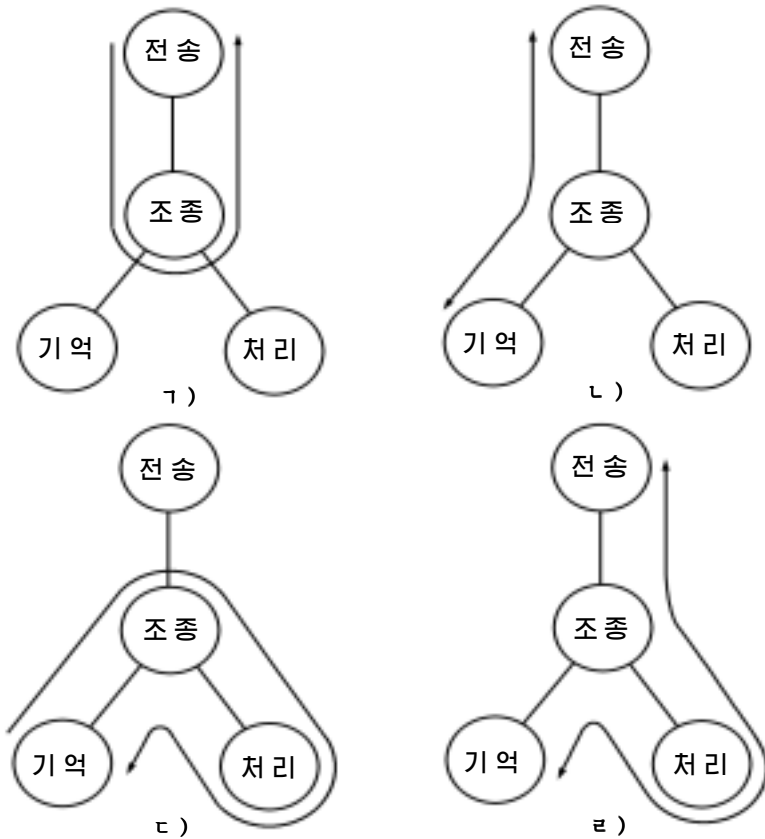



그림 1-2. 가능한 콤퓨터조작

콤퓨터는 자기자신과 다른 모든 장치들사이에 **자료를 전송**할수 있어야 한다. 콤퓨터의 연산환경은 자료의 출발지나 목적지에 봉사하는 장치들로 구성된다. 자료가 콤퓨터에 직접 련결된 장치로부터 수신되거나 송신되는 과정을 **입출력**(I/O)이라고 하며 이 장치를 **주변장치**라고 한다. 자료가 원격장치로부터 또는 원격장치까지 먼 거리를 이동할 때의 과정을 **자료통신**이라고 한다.

마지막으로 이 3가지 기능들을 조종하여야 한다. 결국 이 조종은 명령으로 콤퓨터를 동작시키는 개체들에 의하여 실현된다. 콤퓨터체계내에서 조종장치는 콤퓨터의 자원을 관리하며 명령들에 대응하는 기능적부분들의 성능을 규정한다.

일반적인 준위에 대한 론의에서처럼 가능한 조작의 수는 매우 적다. 그림 1-2에서는 가능한 조작의 4 가지 형태를 그림으로 묘사하였다. 콤퓨터는 단순히 어떤 주변장치로부터 혹은 통신선로로부터 다른 장치에로 자료를 전송하는 전송장치로서의 기능을 수행할수 있다(그림 1-2 ㄱ). 또는 외부장치들로부터 콤퓨터에 전송된 자료를 기억하고 반대로 기억된 자료를 콤퓨터로부터 외부장치에로 출력하는 자료기억기로서의 기능을 수행할수 있다(그림 1-2 ㄴ). 마지막두개의 그림은 기억장치로부터의 자료에 대하여(그림 1-2 ㄷ) 혹

은 기억장치와 외부장치들사이에서의 자료에 대한(그림 1-2 ㄹ) 자료처리를 포함하는 조작들을 보여 준다.

우에서 진행한 론의는 불합리하게 개괄된듯 하다. 콤퓨터구조의 웃준위에서 다양한 기능들을 구별하는데는 확실히 가능하지만 [SIEW82]에서 인용한 다음의 문장에서는 그렇지 않다. 즉 수행되는 기능을 적합하게 하는 주목할 정도로 작은 형태의 콤퓨터구조가 있다. 이 근거에는 모든 기능적인 제한이 프로그람작성시간에 관계되며 설계시간에는 무관계한 콤퓨터의 일반적인 본질이 놓여 있다.

2. 구조

그림 1-3 은 콤퓨터를 가장 단순하게 묘사한것이다. 콤퓨터는 외부장치와 어떤 규약으로 서로 작용하는 실체이다. 일반적으로 외부장치에 대한 결합의 전부는 주변장치와 통신선로로 나눌수 있다. 여기서는 이 두가지 결합형태를 고찰한다.

그러나 이 책에서 크게 관계되는것은 그림 1-4 에서 높은 준위로 보여 준 콤퓨터자체의 내부구조이다. 콤퓨터는 4 개의 기본구성부분을 가진다.

- **중앙처리장치(CPU)**:콤퓨터의 연산을 조종하거나 그의 자료처리기능을 수행한다. 흔히 간단하게 처리장치라고도 한다.
- **주기억기** : 자료를 기억한다.
- I/O : 콤퓨터와 외부장치사이에 자료를 전송한다.
- **체계호상결합**:CPU, 주기억기, I/O 들사이에 통신을 제공하는 기구들

그림 1-3. 콤퓨터

콤퓨터에는 이 구성부분들이 하나이상 있어야 한다. 전통적으로 콤퓨터는 한개의 CPU 를 가진다. 최근에는 단일체계에서 다중처리장치들이 많이 리용되고 있다.

다중처리장치와 관련된 일부 설계계획들을 뚜렷하게 찾아 볼수 있으며 이와 같은 체계는 제 16 장에서 취급한다.이 개별적인 구성부분들은 제 2 편에서 구체적으로 설명한다. 그러나 가장 흥미 있고 가장 복잡한 구성부분인 CPU 의 구조를 그림 1-5 에 주었다. CPU 의 기본구조적구성부분은 다음과 같다.

- 조종장치 : CPU 연산을 조종하며 더 나아가서 콤퓨터를 조종한다.
- 산수론리연산장치 : 콤퓨터의 자료처리기능을 수행한다.
- 등록기 : CPU 에서 내부적기억을 진행한다.
- CPU 호상결합 : 조종장치, ALU 등록기들사이에 통신을 보장하는 기구들이다.

이 구성부분들의 개별적요소들에 대해서는 제 3 편에서 상세하게 설명하였다. 조종장치의 실현을 위한 여러가지 방법이 있지만 가장 일반적인것은 마이크로프로그람조종방식으로 실현하는것이다. 이 방법에서 조종장치의 구조는 그림1-6에서처럼 주어 질수 있다. 이 구조는 제 4 편에서 취급한다.

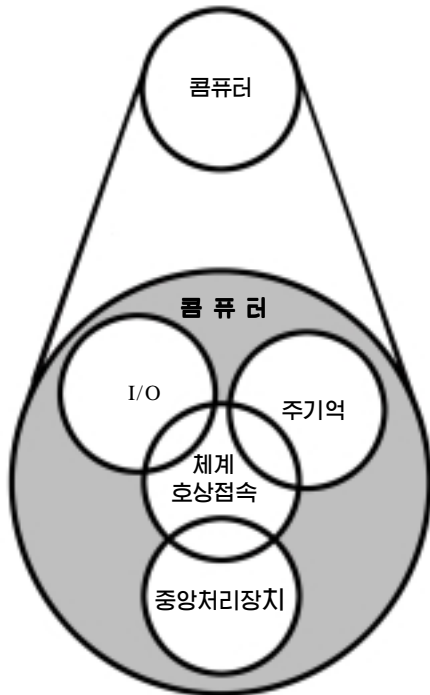



그림 1-4. 콤퓨터의 높은 준위구조

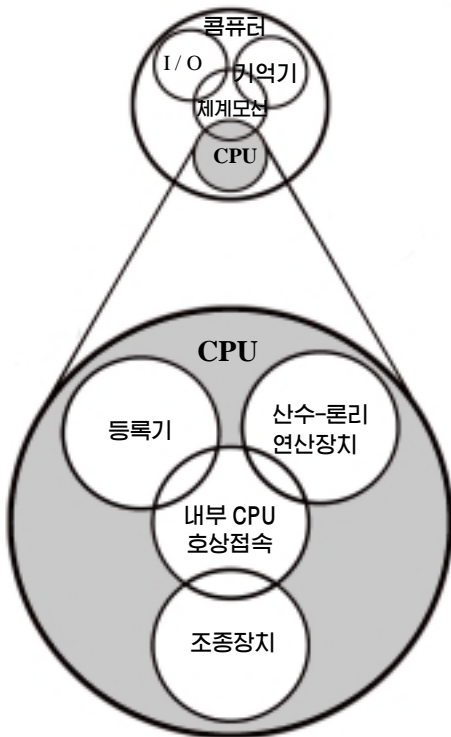



그림 1-5. 중앙처리장치(**CPU**)

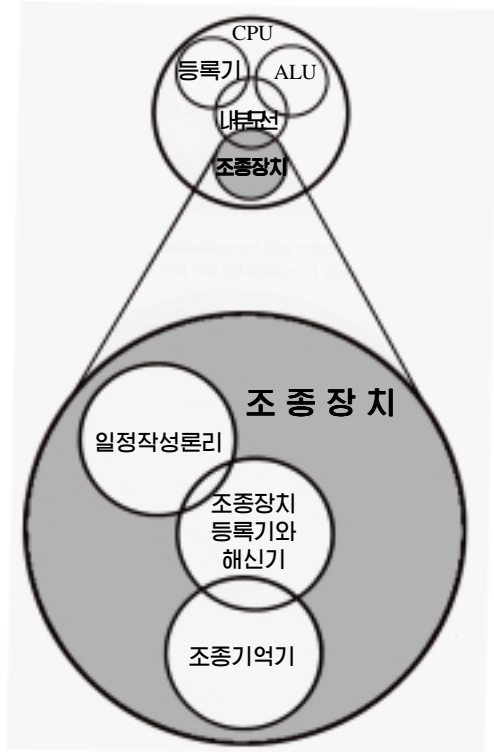



그림 1-6. 조종장치

제 3 절. 책의 개요

이 절에서는 책의 전반내용을 소개한다. 뒤에서 취급하는 장들의 간단한 개요는 다음과 같다.

콤퓨터의 발전과 성능

제 2 장은 두가지 내용을 담고 있다. 첫째로, 콤퓨터조직과 구성방식의 기본개념을 소개하는 흥미 있는 콤퓨터의 발전력사이다. 둘째로, 성능을 높이기 위한 성공적인 콤퓨터체계의 설계요점들에 대한 기술발전추세와 균형, 효률적인 성능을 달성하기 위하여 리용되는 여러가지 기술과 전략을 미리 고찰하는것이다.

체계모선

제일 웃준위에서 콤퓨터는 처리장치, 기억기, I/O 구성부분들로 구성된다. 콤퓨터의 기능적인 동작은 구성부분들사이의 자료와 조종신호들의 교환으로 이루어 진다. 자료교환을 실현하기 위하여서는 이 구성부분들이 서로 련결되여야 한다. 제 3 장에서는 콤퓨터의 구성부분들과 입출력요구들에 대하여 간단히 설명하였다. 호상결합설계에 영향을 미치는 관건적문제들과 새치기를 실현하기 위한 요구들을 고찰하였다. 이 장의 기본은 모선구조의 리용과 같은 호상결합에 대한 가장 일반적인 방법을 학습하는것이다.

내부기억기

콤퓨터기억기는 형태, 기술, 구성, 성능, 가격의 넓은 범위를 포괄한다. 일반적인 콤퓨터체계는 일부 내부기억기(처리장치에 의하여 직접 호출가능한기억기)와 일부 외부기억기(처리장치가 I/O 모듈을 통하여 호출하는 기억기)와 같은 기억기보조체계들의 계층으로 구성된다. 제 4 장에서는 이 계층과 내부기억기에 관계되는 설계의 요점들에 대하여 고찰하였다. 우선 반도체주기억기의 본질과 구성이 설명되고 다음으로 명령캐쉬와 자료캐쉬의 분할방식과 그 준위캐쉬를 포함한 캐쉬기억기설계의 상세한 내용을 주었다. 마지막부분에서 최근 개량된 DRAM 기억기구성들을 소개하였다.

외부기억기

제 5 장에서는 자기디스크기억기와 관련된 여러가지 설계와 성능지표들을 설명하였다. 그리고 점차 일반화되고 있는 RAID 방식을 설명하였다. 또한 빛기억기와 자기테프체계를 설명하였다.

입출력

I/O 모듈은 처리장치와 기억기 그리고 매개 조종장치들 혹은 대부분 외부장치들과 서로 련결된다. 제 6 장에서는 프로그람식 I/O 조종방식, 새치기 I/O 조종방식, 직접기억기접근(DMA)조종방식의 기술을 리용하여 I/O 모듈이 콤퓨터체계를 안정하게 서로 작용할수 있는 기구들에 대하여 고찰하였다. 그리고 I/O 모듈과 외부장치들사이의 결합방식을 서술하였다.

조작체계지원

이 문제에서는 기본콤퓨터구성부분들이 효과적인 작업을 수행하기 위하여 어떻게 관리되며 콤퓨터장치가 조작체계지원을 제공하기 위하여 어떻게 구성되는가를 설명하기 위하여 조작체계를 기본으로 고찰한다. 제7장은 조작체계들의 기본형태들을 일치시키고 그것들을 사용하게 된 간단한 력사로부터 시작한다. 그다음 다중프로그람방식을 긴시간일정작성과 짧은시간일정작성기능을 조사하는것으로 설명한다. 끝으로 기억기관리에서는 토막화, 페지화, 가상기억기를 론의한다.

콤퓨터산수연산

제8장에서는 콤퓨터산수연산의 론의로서 처리장치를 상세히 고찰한다. 처리장치들은 일반적으로 옹근수 혹은 고점수와 류점수의 두가지 형태를 제공하고 있다. 두 경우에 대하여 여기서는 처음에 수들의 표현을 고찰하고 그다음 산수연산방식을 론의한다. 또한 중요한 IEEE754 류점수의 표준화를 상세히 고찰한다.

명령모임

프로그람작성자의 견지에서 처리장치의 연산을 리해하는 가장 좋은 방법은 처리장치가 실행하는 기계어명령모임을 배우는것이다. 제9장에서는 기계어명령모임들의 기본특성들을 고찰한다. 처리장치는 명령모임에서 찾아 볼수 있는 여러가지 자료형식과 연산형태들을 수행할수 있게 구성된다. 다음으로 기호언어와 처리소자명령들사이의 관계를 간단히 설명하였다. 제 10 장에서는 가능한 주소화방식들이 소개되였으며 또한 상품화의 내용들을 포함하여 명령형식의 지표들을 소개하였다.

CPU 구조와 기능

제 11 장에서는 처리장치의 내부구조와 기능에 대한 내용을 주었다. 전반적인 구성(ALU , 조종부, 등록기파일)이 다시 고찰된다. 다음으로 등록기파일의 구성이 론의된다. 이 장의 뒤부분에서 기계어명령을 실행하기 위한 처리장치의 기능화를 서술한다. 명령주기는 명령꺼내기, 간접처리, 실행, 새치기주기들의 기능과 호상련관을 고찰한다. 마지막으로 성능을 높이기 위한 관흐름조종의 리용을 상세히 소개한다.

축소명령모임콤퓨터

최근에 콤퓨터조직과 구성방식에서 가장 큰 혁신은 축소명령모임콤퓨터(RISC)방식이다. RISC 구성방식은 처리소자방식에서 력사적추세로부터의 극적인 새로운 발전이다. 이 연구방법에 대한 해석은 콤퓨터조직과 구성방식에서 중요한 대부분의 지표들을 대상으로 하여 진행된다. 제 12 장에서는 RISC 방식을 기본으로 하고 복합명령모임콤퓨터(CISC)방식과 비교한다.

명령준위병렬화와 슈퍼스칼라처리장치

제 13 장에서는 가장 최근까지도 중요한 설계혁신이라고 하는 슈퍼스칼라처리장치를 고찰한다. 비록 슈퍼스칼라기술이 임의의 처리장치에서 리용될수 있지만 RISC 방식에 특

별히 더 적합하다. 또한 명령준위병렬화의 일반적인 지표들을 고찰한다.

조종부연산

제 14 장에서는 처리장치기술들이 조종부의 기구들에 의하여 어떻게 수행되며 더 특수하게 처리장치의 여러 구성부분들이 이 기능들을 수행하기 위하여 어떻게 조종되는가에 대하여 론의한다. 매개 명령주기는 조종신호들을 형성하는 마이크로연산묶음의 조합이다. 실행은 조종장치로부터 ALU, 등록기, 체계호상결합구조까지 출력되는 조종신호들의 작용으로 완료된다. 끝으로 장치적실현에 의한 조종부의 실현방법을 주었다.

마이크로프로그람조종방식

제 15 장에서는 조종부가 마이크로프로그람방식의 기술을 리용하여 어떻게 실현되는가를 보여 준다. 우선 마이크로조작들은 마이크로명령으로 배렬된다. 그다음 매개 기계어명령을 위한 마이크로프로그람을 포함하고 있는 조종기억기설계방안을 서술하였다. 또한 마이크로프로그람방식의 조종부구조와 기능을 설명하였다.

병렬처리

전통적으로 콤퓨터는 순서기계로 고찰되여 왔다. 콤퓨터기술이 발전되고 콤퓨터장치의 가격이 눅어 짐에 따라 콤퓨터설계가들은 보통 성능을 높이고 믿음성을 높이기 위하여 점차적으로 병렬처리방법들을 연구하고 있다. 제 16 장은 병렬처리조직을 위한 여러가지 방법들을 보여 준다. 또한 다중처리장치방식을 위하여 캐쉬일치성의 기본설계지표를 고찰한다.

수자론리

이 책에서 기본은 콤퓨터체계의 기초적인 블로크들로서 2 진기억요소와 수자기능들을 취급한다. 이 책의 부록은 이 기억요소들과 기능들이 수자론리로 어떻게 실현되는가를 서술하였다. 다음으로 문의 개념을 소개하였다. 마지막으로 문들로부터 구성되는 조합회로와 순서회로들을 론의하였다.

제 4 절. 인터네트 및 Web 자원

이 책에는 이 분야를 계속 발전시키는데 도움을 주는 인터네트와 여러가지 Web 자원들이 있다.

이 책의 Web 싸이트

이 책에 지적된 Web 페지는 http://www.shore.net/~ws/COA5e.html 에 설치하였다.

이 Web 싸이트에 대한 구체적인것은 이 책의 앞부분을 보면 된다.

즉시로 임의의 편집이나 다른 오유들을 발견하면 이 책의 오유목록은

http://www.shore.net/~ws 에서 볼수 있다. 이 파일들은 요구대로 갱신될수 있다. ws@shore_net 에 반점을 찍어서 임의의 오유를 전자우편으로 보여 주어야 한다. 다른 책들에 대한 고침표들은 참작순서정보와 같이 Web싸이트에 있다.

다른 Web싸이트

이 책의 제목들과 관련된 일부 짧은 정보들을 제공하는 여러가지 Web 싸이트들이 있다. 보조장들에서 지적된 Web 사이트들에 대한 지적자들은 {참고문헌}에서 찾아 볼수 있다. Web싸이트들에 대한 URL 들은 자주 변하는 경향성을 가지므로 이 책에 그것들을 포함하지 않는다. 이 책에서 소개된 전체 Web싸이트들에 대하여 해당한 관련부분들은 이 책의 Web싸이트에서 찾을수 있다.

다음의것들은 콤퓨터조직과 구성방식과 관련하여 일반적으로 흥미를 가지는 Web 싸이트들이다.

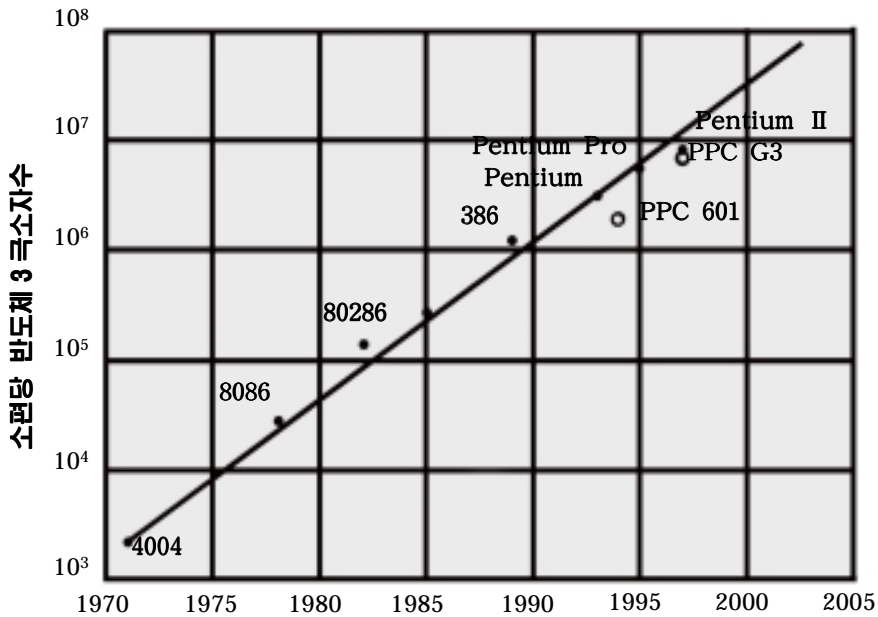
- WWW Computer Architecture Home Page: 콤퓨터구성방식그룹들과 계획, 구성기술, 문헌과 사용방법, 상업적정보들을 포함하는 콤퓨터구성방식연구자들과 관련된 정보들에 대한 포괄적인 색인
- CPU Info Center: 기술적론문, 제품정보, 제일 마지막으로 발표된 번호를 포함하는 특수한 처리장치들의 정보
- ACM Special Interest Group On Cputer Architecture: SIGARCH 활용들과 발표에 대한 정보
- IEEE Technical Committee on Cputer Architecture: TCAA 에 시사통보의 복사
- Intel Technology Journal: 인텔의 직결발표

USENET 새로운 그룹

여러 USENET 그룹들은 콤퓨터조직과 구성방식의 몇가지 형태를 위하여 노력하고 있다. 실제로 모든 USENET 그룹들에 있어서 신호 대 잡음비가 높으나 요구를 만족시킨다면 실험할 가치가 있다. 기본적으로 관련되는것은 다음과 같다.

- **Comp. Arch**: 콤퓨터구성방식의 론의를 위한 일반적인 그룹. 흔히 좋게 평과됨
- **Comp. Arch arithmetic**: 콤퓨터산수연산알고리듬과 표준화들의 론의
- **Comp. Arch storage**: 기술적인 문제들로부터 실제적인 사용지표들에 이르기까지 범위들을 론의

제 2 장. 콤퓨터의 발전과 성능

◆◆

- ◆ 콤퓨터가 얼마나 발전하겠는가는 처리장치속도가 얼마나 빠르고 구성부분크기가 어느 정도 작으며 또 기억용량이 얼마나 크고 I/O 능력과 속도가 얼마나 높은가에 의해 특징 지어 진다.
- ◆ 처리장치의 속도를 높이는데 관계되는 요인의 하나는 극소형처리장치의 구성부분들사이의 거리이다. 즉 요소들사이의 거리가 짧아 지면 결국 속도가 높아 진다. 그러나 최근년간 속도에서의 실제적인 리득은 관흐름조종을 확고히 리용하고 앞의 명령의 림시적실행결과로 나타나는 필수적인 투기명령기술을 리용하는 처리장치의 구성과 관련된다. 이 모든 기술에 의하여 처리장치가 항시적으로 동작하도록 설계된다.
- ◆ 콤퓨터체계설계에서 중요한것은 한 부분의 성능을 높였을 때 다른 부분의 지연으로 인한 장애를 없애기 위하여 여러가지 구성부분들의 성능의 균형을 유지하는것이다. 특히 처리장치속도는 기억기접근속도보다 더 빨리 증가한다. 이 불균형적인 구성을 균형화하기 위하여 캐쉬의 리용, 기억기로부터 처리장치까지의 넓은 자료통로의 리용, 더 리상적인 기억기소편의 리용과 같은 다양한 기술을 적용한다.

◆◆

콤퓨터의 간단한 력사를 보기로 하자. 콤퓨터력사는 그자체가 흥미 있으며 콤퓨터구성과 기능에 대한 고찰방법을 준다. 다음으로 성능지표들을 론의한다. 이 책에서는 콤퓨터자원들을 균형적으로 리용하는데 필요한 요구를 충분히 서술하였다. 끝으로 Pentium 과 PowerPC 를 기본실례로 두 체계의 발전과정을 간단히 보여 준다.

제 1 절. 콤퓨터의 간단한 력사

1. 제 1 세대: 전자관식콤퓨터

ENIAC(Electronic Numerical Integrator And Computer)

펜실바니아종합대학의 죤 프레스퍼 에커트와 죤 머클리에 의하여 설계되고 그들의 감독밑에 제작된 ENIAC 는 세계에서 첫 전자관식수자콤퓨터였다.

개발계획은 전시의 요구들에 대응한것이였다. 새로운 무기의 개발방향과 탄도표에 대하여 책임지고 있는 대리소인 미군탄도학연구소(BRL)는 제한된 시간내에 이 표들을 정확하게 확증하기가 매우 곤난했다. 이 발사표들이 없으면 포수들에게는 새로운 무기와 대포들이 쓸모 없었다. BRL 은 탁상형수산기를 리용하는 200 명이상의 성원들을 채용하여 필요한 탄도방정식을 풀었다. 한개의 무기에 대한 표를 준비하자면 한사람이 여러날동안에 많은 시간을 소비하여야 하였다.

펜실바니아종합대학의 전자공학 교수인 머클리와 그의 졸업반 학생인 에커트는 BRL 에 응용할수 있는 전자관을 리용한 범용콤퓨터를 제작할것을 계획하였다. 1943 년에 군부에서 이 제안을 받아 들였으며 ENIAC 로 작업이 시작되였다. 만들어 진 콤퓨터는 30t 의 질량과 15,000 평방피트의 면적을 차지하며 18000 개이상의 전자관을 가진 대단히 큰 기계였다. 연산할 때 140KW 의 전력을 소비하였다. 또한 이미 있던 전자기계식콤퓨터보다 연산속도가 근본적으로 빠르며 초당 5000 회의 더하기연산능력을 가지고 있었다.

ENIAC 는 2 진수콤퓨터가 아니라 10 진수콤퓨터이다. 즉 수들은 10 진수형식으로 표현되며 산수연산은 10 진수체계로 수행된다. 기억기는 20 개의 축적기로 구성되였으며 그 매개는 10 개의 10 진수자리를 등록할수 있다. 매 수자는 10 개의 전자관의 직렬련결에 의하여 표현된다. 임의의 시간에 한개의 전자관이 ON 상태에 있으면 10 개의 수자중 하나를 표현한다. ENIAC의 기본결함은 스위치를 설정하여 케블을 련결, 차단하면서 수동으로 프로그람을 작성하는것이다.

ENIAC 는 1946 년에 완성되였으며 2 차대전에서 효력을 내기에는 너무 늦었다. 대신에 첫 과제는 수소탄의 가능성을 측정하기 위한 복잡한 계산들을 실행하는것이였다. ENIAC 는 그것이 해체될 때인 1955 년까지 BRL 의 관리하에 연산을 계속하였다.

폰 노이만형콤퓨터

ENIAC 를 위한 프로그람입력과 변환작업은 매우 오랜시간이 걸렸다. 만일 프로그람이 자료와 나란히 기억기에 기억될수 있는 적당한 형식으로 표현될수 있다면 프로그람 작성과정이 쉬울것이다. 그리고 콤퓨터는 기억기로부터 그것들을 읽는것으로 명령을 얻을것이며 프로그람은 기억기의 위치값을 설정하는것으로서 설정되고 변환될것이다.

프로그람기억식개념으로 알려 진 이 구상은 일반적으로 ENIAC 설계자들에 의하여 시작되였으며 가장 명백하게는 ENIAC제작시에 고문이였던 죤 폰 노이만에 의하여 개발되였다. 또한 이 구상은 튜링에 의하여 같은 시각에 개발되였다. 구상은 새로운 콤퓨터 EDVAC 를 위하여 노이만이 1945 년에 처음으로 발표하였다.

1946 년에 노이만과 동업자들은 IAS 콤퓨터라는 새로운 프로그람기억식콤퓨터의 설계를 시작하였다. IAS 콤퓨터는 1952 년까지 완성되지 못하였지만 그후의 모든 범용콤퓨터들의 원형으로 되였다.

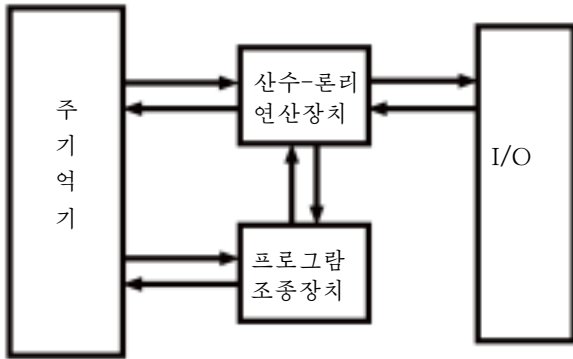



그림 2-1. IAS 콤퓨터의 구성

그림 2-1 은 IAS 콤퓨터의 일반구조를 보여 준다. 콤퓨터는 다음과 같이 구성되였다.

- 주기억기 : 자료와 명령을 기억한다.
- 산수-론리연산부(ALU) : 2 진자료에 대한 연산을 수행한다.
- 조종부 : 기억기로부터의 명령을 해석하고 그것들을 실행하도록 한다.
- I/O 장치 : 조종부에 의하여 조작된다.

이 구조는 일찌기 노이만의 제안으로부터 나왔는데 여기서 간단히 인용하겠다[VONN45].

2.2 **첫째로**, 장치는 기본상 계산기이므로 자주 산수적인 기본연산을 수행하여야 할것이다. 이것들은 더하기, 덜기, 곱하기, 나누기이다. 그러므로 이 연산들을 위한 특수한 기구들을 포함하여야 하는것은 응당하다. 그러나 이 연산들에 대한 원리는 일반적으로만 알려 져 있으며 관련되는 특수한 방법들을 더 정밀하게 조사할것을 요구한다. 적어도 **중앙산수연산**부분이 존재하여야 할것이며 이것은 **첫번째로 지적된 부분**인 CA 를 구성한다.

2.3 **둘째로**, 장치의 론리적인 조종 즉 연산의 알맞는 순서화는 중앙조종장치구성에 의해서만 수행될수 있다. 장치가 융통성이 있다면 즉 모든 목적에 될수록 밀접하다면 구별은 개별적인 문제를 정의하고 해결하기 위하여 지적되는 명령들과 이 명령들을 수행한다고 볼수 있는 일반적인 조종기구들사이에 이루어 진다. 명령들은 몇가지 방법으로 기억되여야 한다. 조종장치들은 장치의 일정한 조작부분에 의하여 표현되여야 한다. **중앙조종**장치는 이 조종장치의 기능뿐아니라 **두번째로 지적된 부분** CC 를 수행하도록 하는 기구를 의미한다.

2.4 **셋째로**, 길고 복잡한 조작(계산의 특수성)들의 렬을 수행하는 장치는 상당한 기억기를 가져야 한다.

(b)복잡하게 얽힌 문제를 해결하는 명령들은 특히 코드가 세부적이라면 상당한 기능적인 명령들로 이루어 져야 한다. 이 기능적인 명령은 기억되여야 한다.

적어도 모든 **기억기**는 **셋째로 지적된 장치부분** M 으로 구성된다.

2.6 CA, CC, M 으로 지적된 3 개의 부분들은 인간의 신경계통과 관련되는 신경세포에 대응된다. 그것은 감각이나 신경, 운동신경이나 배출하는 신경세포의 동등성을 여전히 론의한다. 이것들은 장치의 입력, 출력기구들이다. 장치는 이 류형의 일부 지적된 매체들과 련결된 입력, 출력을 유지하는 능력이 있어야 한다. 매체는 장치 R 의 **외부기록매체**라고 부른다.

2.7 **넷째로**, 장치는 R 에서 특수부분 C 와 M 에 정보를 전송하는 기구들을 가져야 한다. 이 기구들은 **입력**, **넷째로 지적된 부분** I 를 형성한다. 그것은 R 로부터 M 에로, C 로부터 직접적으로 모든 전송을 하게 하는데 가장 좋은것이라는것을 알수 있다.

2.8 **다섯째로**, 장치는 지적된 부분 C 와 M 에서 R 로 전송하는 기구들을 가져야 한다. 이 기구들은 **출력** 즉 **다섯번째로 지적된 부분** O 를 형성한다. 그것은 M 으로부터(O 에 의하여) R 에로, C 로부터 직접적으로 모든 전송을 하게 하는데 가장 좋은것이라는것을 알수 있다.

아주 드문 경우를 내놓고는 오늘날 모든 콤퓨터들은 이와 같은 일반구조와 기능을 가지며 따라서 노이만형콤퓨터라고 한다. 그러므로 IAS 콤퓨터[BURK46]의 연산기능에 대하여 간단히 서술할 필요가 있다. 다음의 참고문헌[HAYE88]에서 폰 노이만의 용어와 표기들은 현재표기법들과 일치하도록 변경시켰다. 즉 이 론의를 첨부한 실례와 그림 설명들은 뒤에 취급된 본문들에 기초하였다.

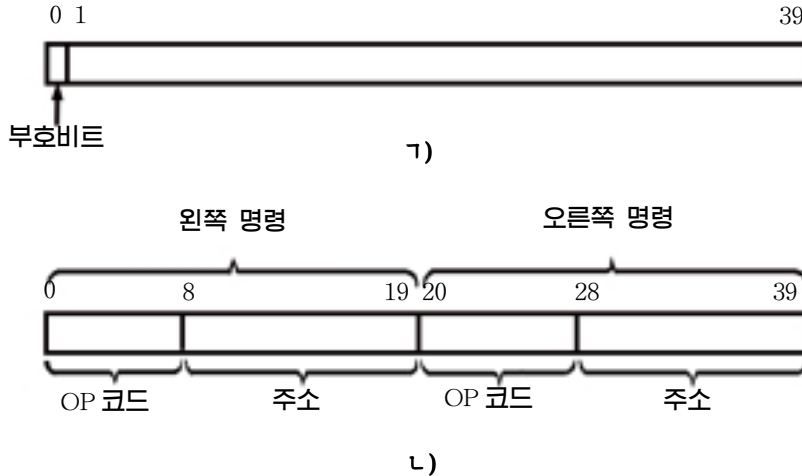



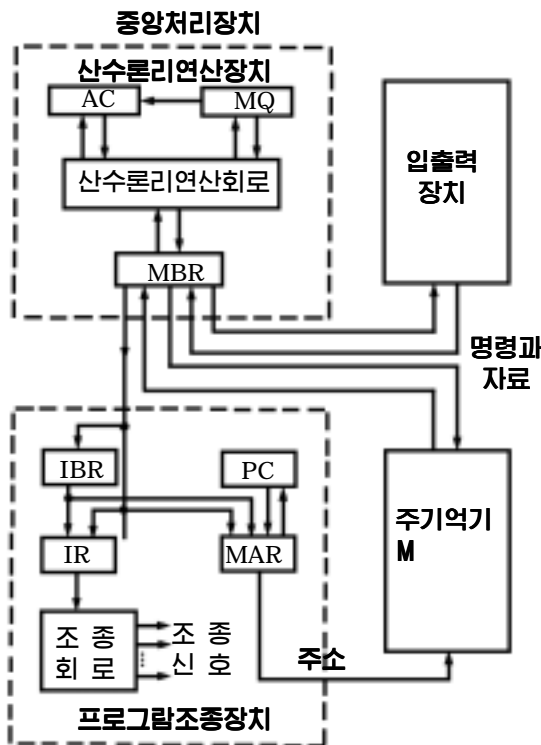
그림 2-2. IAS 기억기형식

ㄱ- 수값단어, ㄴ-명령단어

IAS 의 기억기는 40bit 로 구성된 **단어**를 기억시킬수 있는 1000 개의 기억장소로 구성되였다. 자료와 명령들은 여기에 기억된다. 그러므로 수들은 2 진수형식으로 표현되여야 하며 매개 명령들은 2 진코드여야 한다. 그림 2-2 에서는 이 형식들을 그림으로 설명하였다.

그림 2-3. IAS 콤퓨터의 확장된 구조

매개 수자는 한개의 부호비트와 39bit 의 값으로 표현된다. 한개의 단어는 2 개의 20 bit 명령을 포함하며 매개 명령은 수행할수 있는 연산을 지정하는 8bit 의 조작코드와 기억기의 단어를 지적하는 12bit 의 주소로 구성된다(0 부터 999 까지의 번호).

조종부는 기억기로부터 명령을 꺼내고 주어 진 시간에 그것을 실행하도록 IAS 를 조작한다. 이 실례에 대한 더 상세한 구조블로크도를 그림 2-3 에 보여 주었다. 이 그림은 조종부와 ALU 가 등록기를 포함한다는것을 보여 주고 있다. 등록기에 대한 구체적내용은 다음과 같다.

- **기억완충등록기(MBR) :** 기억기에 기억하기 위한 단어를 보관하거나 기억기로부터 단어를 받기 위하여 리용된다.
- **기억주소등록기(MAR):** MBR 에 단어를 읽어 넣거나 MBR 로부터 기억기에 단어를 쓰기 위한 기억기의 주소를 지적한다.

- **명령등록기 (IR)** : 실행되고 있는 8bit 조작코드를 등록한다.
- **명령완충등록기(IBR)** : 기억기의 단어로부터 오른쪽 명령을 림시적으로 유지하기 위하여 리용된다.
- **프로그람계수기(PC)** : 기억기로부터 꺼내여 지는 다음명령쌍의 주소를 등록한다.
- **축적기(AC)와 곱하는수/상등록기(MQ)** : ALU 연산의 연산수들과 결과들을 림시적으로 보관하기 위하여 사용한다. 실례로 두개의 40bit 수들을 곱한 결과는 80bit 수이며 웃부분의 40bit 는 AC 에 기억되고 아래부분의 40bit 는 MQ 에 기억된다.

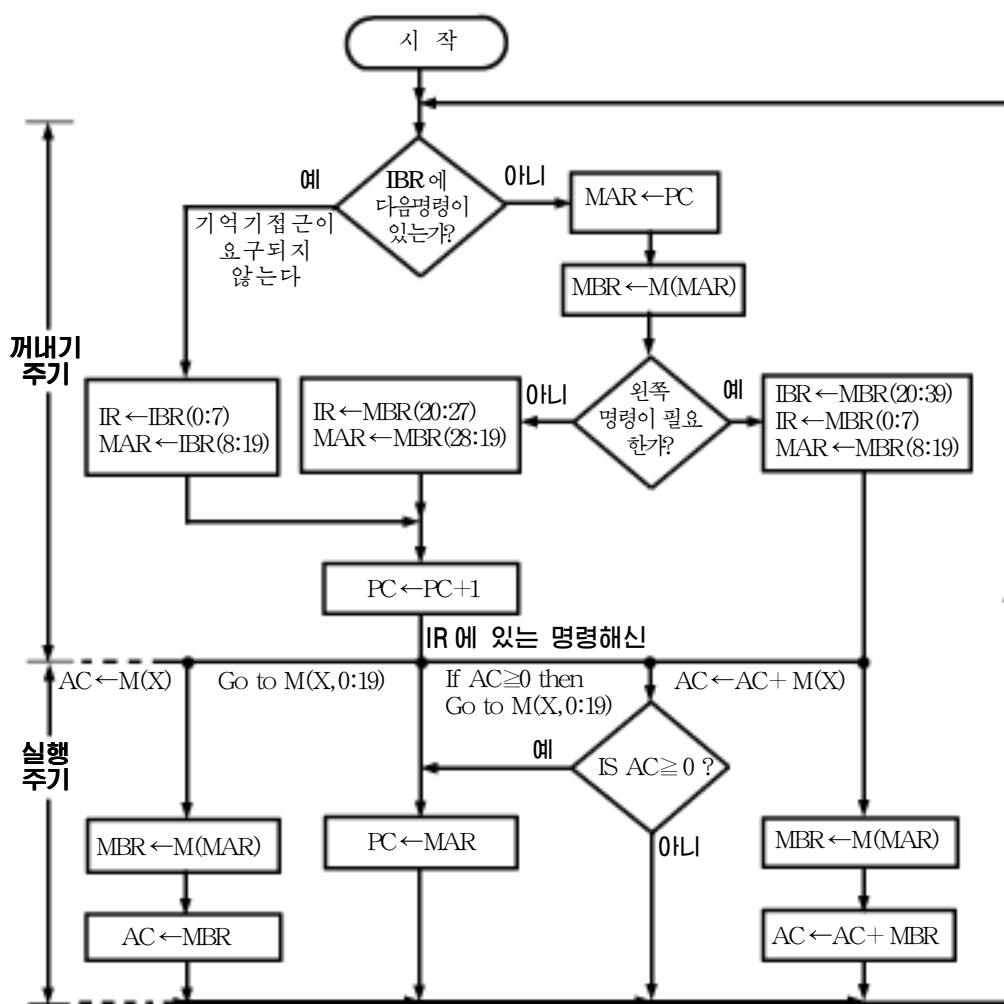
IAS 는 그림 2-4 에서 보여 주는바와 같이 **명령주기**실행을 반복하는 방법으로 연산한다. 매 명령주기는 두개의 보조주기로 이루어 진다. **꺼내기주기**를 실행하면 다음명령의 조작코드는 IR 에 넣어 지고 주소부는 MAR 에 넣어 진다. 이 명령은 IBR 로부터 꺼내여 질수도 있으며 혹은 MBR 로부터 단어읽기에 의하여 기억기로부터 주어 질수도 있는데

M(X) = 주소가 X 인 기억위치의 내용
(X:Y) = X 비트들과 Y 비트들

그림 2-4 . IAS 연산의 부분흐름도

이때에는 BR, IR, MAR에 넣어 진다.

왜 간접적방법이라고 하는가. 이 조작들은 전자회로적으로 조종되며 자료통로를 리용하여 실현된다. 전자회로를 간단히 하기 위하여 읽기나 쓰기를 위한 기억기주소를 지적하는데 리용되는 등록기가 한개 있으며 원천 혹은 목적을 위하여 리용되는 등록기도 한개 있다.

일단 조작코드가 IR에 입력되면 실행주기가 수행된다. 조종회로는 조작코드를 해신하며 전송되는 자료를 발생시키거나 ALU에 의하여 수행되는 연산을 위한 적당한 조종신호를 출력하는것으로 명령을 실행한다.

IAS콤퓨터는 표 2-1에 서술된것처럼 모두 21개의 명령을 가진다.

표 2-1. IAS명령모임

명령형태	조작코드	기호언어	해설
자료전송	00001010	LOAD MQ	MQ등록기의 내용을 AC축적기에 전송
	00001001	LOAD MQ,M(X)	기억주소 X의 내용을 MQ에 전송
	00100001	STOR M(X)	축적기의 내용을 기억주소 X에 전송
	00000001	LOAD M(X)	M(X)를 축적기에 전송
	00000010	LOAD –M(X)	- M(X)를 축적기에 전송
	00000011	LOAD [M(X)]	M(X)의 절대값을 축적기에 전송
	00000100	LOAD –[M(X)]	-[M(X)]를 축적기에 전송
무조건분기	00001101	JUMP M(X,0:19)	M(X)의 왼쪽 절반위치로부터 다음명령에로 이행
	00001110	JUMP M(X,20:39)	M(X)의 오른쪽 절반위치로부터 다음명령에로 이행
조건분기	00001111	JUMP +M(X,0:19)	축적기의 값이 정수이라면 M(X)의 왼쪽 절반위치로부터 다음명령에로 이행
	00010000	JUMP +M(X,20:39)	축적기의 값이 정수이라면 M(X)의 오른쪽 절반위치로부터 다음명령에로 이행
산수연산	00000101	ADD M(X)	M(X)를 AC에 더하기:AC의 결과를 넣기
	00000111	ADD [M(X)]	[M(X)]를 AC에 더하기:AC의 결과를 넣기
	00000110	SUB M(X)	AC로부터 M(X)를 덜기:AC의 결과를 넣기
	00001000	SUB [M(X)]	AC로부터 [M(X)]를 덜기:AC의 결과를 넣기
	00001011	MUL M(X)	M(X)에 MQ를 곱하기:AC에 웃부분결과를 넣기,MQ에 아래부분결과를 넣기
	00001100	DIV M(X)	AC를 M(X)로 나누기: M(Q)에 상을 넣기,AC에 나머지
	00010100	LSH	축적기를 2배하기(왼쪽으로 한비트밀기)
	00010101	RSH	축적기를 2로 나누기(오른쪽으로 한비트밀기)
주소변경	00010010	STOR M(X,8:19)	M(X)의 왼쪽 주소마당을 AC의 오른쪽 12bit로 바꾸기
	00010011	STOR M(X,28:39)	M(X)의 오른쪽 주소마당을 AC의 오른쪽 12bit로 바꾸기

이 명령들은 다음과 같이 분류할수 있다.

- **자료전송**: 기억기와 ALU 등록기사이 혹은 두 ALU 등록기들사이 자료를 전송한다.
- **무조건이행**: 일반적으로 조종부는 기억기로부터 순서대로 명령을 실행한다. 이 순서는 분기명령에 의하여 변화될수 있다. 이것은 반복적인 연산을 쉽게 한다.
- **조건이행**: 분기는 분기점을 허락하는 조건에 의거하여 이루어 진다.
- **산수연산**: ALU 에 의하여 수행되는 연산이다.
- **주소변경**: ALU 에서 계산하도록 주소를 허락하며 이 주소는 기억기에 기억된 명령들에 삽입된다. 이것은 프로그람으로 하여금 가능한 주소화의 유연성을 허락한다.

표 2-1 에 쉽게 읽을수 있는 형식으로 기호언어명령을 주었다. 사실상 매 명령은 그림 2-2 ㄱ의 형식으로 구성되여야 한다. 조작코드부(첫 8bit)는 실행될수 있는 21 개 명령중의 임의의것을 지정한다. 주소부(나머지 12bit)는 명령의 실행에 포함되는 1000 개의 기억주소들중의 하나를 지정한다.

그림 2-4 에는 조종부에 의한 여러가지 명령실행의 실례를 보여 주었다. 매개 명령은 여러 단계를 요구한다고 보자. 이것들의 일부는 아주 잘 만들었다. 곱하기연산은 39 개의 보조연산들을 요구하며 그중 한비트는 부호비트이므로 제외한다.

콤퓨터의 상품화

1950 년대는 시장을 독점한 Sperry 와 IBM 두 회사에 의한 콤퓨터산업의 출현을 보여 주었다.

1947 년에 에커트와 머클리는 상품용콤퓨터를 만들기 위한 에커트-머클리콤퓨터회사를 내왔다. 이들이 처음으로 만든 콤퓨터는 1950 년 예산안계산을 위하여 통계조사사무국의 의뢰로 만들어 진 UNIVAC Ⅰ(Universal Automatic Computer)였다. 에커트-머클리회사는 Sperry-Rand 회사의 UNIVAC 부문의 한 부분으로 되였다.

UNIVAC Ⅰ은 처음으로 성공한 상품화된 콤퓨터였다. 이름이 의미하는것처럼 이 콤퓨터는 과학적이며 그리고 실용적인 응용을 목적으로 한것이였다. 이 체계를 서술한 첫 론문에서는 콤퓨터가 수행할수 있는 행렬대수계산, 통계문제, 인체보험회사를 위한 광고, 군사문제들에 대하여 서술하였다.

UNIVAC Ⅰ보다 더 큰 기억기용량과 더 높은 성능을 가진 UNIVAC Ⅱ는 1950 년대 후반기에 개발되였으며 콤퓨터산업의 특징을 나타내는 여러가지 경향성들을 가지고 있다. 첫째로, 기술이 발전함에 따라 회사들로 하여금 더 크고 더 능률적인 콤퓨터를 계속 제작하도록 하였다. 둘째로, 매 회사는 낡은 콤퓨터와 웃방향호환성을 가지는 새로운 콤퓨터를 만들려고 시도하였다. 이것은 낡은 콤퓨터에서 작성된 프로그람이 새로운 콤퓨터에서도 실행될수 있다는것을 의미한다. 이 전략은 사용자의 요구로부터 출발한것이였다. 즉 사용자들이 새로운 콤퓨터를 사려고 할 때에는 프로그람들에 투자하는 손실을 없애기 위하여 같은 회사의것을 사려고 한다.

UNIVAC 는 판매를 목적으로 1100 계렬의 콤퓨터를 개발하기 시작하였다. 계렬이 동시에 존재하는 특징들을 설명하자. 첫 형 UNIVAC1103 과 여러해동안에 생산된 이 계렬의 형들은 기본적으로 길고 복잡한 계산을 요구하는 과학분야에 적용되였다. 다른 회사들의 형들은 많은 량의 본문자료처리를 요구하는 봉사분야의 적용에 집중하였다. 이러한 적용목적의 분리는 크게 나타나지 않았으나 몇년동안에 뚜렷하게 나타났다.

MarK 의 제작을 도와 주었으며 그후 카드처리장치의 기본제작업체인 IBM 은 1953 년에 첫 전자프로그람기억식콤퓨터를 개발하였다. 701 은 기본적으로 과학용으로 리용하였다. 1955 년에 IBM 은 봉사를 목적으로 하는 일부 장치적특징을 가지는 702 제품을 개

발하였다. 이 콤퓨터들은 IBM 을 압도적으로 우세한 콤퓨터제작업체가 만든 첫 700/7000 콤퓨터의 계렬이다.

2. 제 2 세대: 3 극소자콤퓨터

전자관식콤퓨터에서 처음으로 되는 기본변화는 전자관을 3 극소자로 바꾼것이다. 반도체 3 극소자는 전자관보다 훨씬 작고 눅으며 열손실이 적으나 콤퓨터제작은 전자관식과 같은 방법으로 진행되였다. 도선, 금속판, 유리관과 진공을 요구하는 전자관과는 달리 3 극소자는 규소로 만든 **고체소자**이다.

3극소자는 1947년에 벨 라브스에 의하여 발명되였으며 1950년대에 전자혁명이 시작되게 하였다. 그러나 1950 년대 후반기까지 완전히 3 극소자로 제작된 콤퓨터는 없었다. IBM 은 이 새로운 기술을 개발한 첫 회사로 되지 못했다. 더 명백히는 NCR 와 RCA 가 몇개의 작은 3 극소자들로 front-runners 를 만들었다. IBM 은 인차 7000 계렬콤퓨터를 만들었다.

3극소자를 리용하여 만든 콤퓨터를 **제2세대콤퓨터**라고 정의한다. 기초장치기술에 기초하여 세대별로 콤퓨터를 분류할수 있다(표 2-2). 새로운 세대로 바뀔 때마다 전 세대보다 더 높은 속도, 더 큰 기억용량, 더 작은 물리적크기를 가진 콤퓨터가 출현하였다.

표 2-2. 콤퓨터세대

세대	대략적인 기간	기술	일반적인 속도 (초당 연산수)
1	1946～1959	전자관	40,000
2	1958～1964	3 극소자	200,000
3	1965～1971	중소규모집적회로	1,000,000
4	1972 ～1977	대규모집적회로	10,000,000
5	1978～	초대규모집적회로	100,000,000

또한 다른 변화들도 있다. 제 2 세대콤퓨터들에는 더 복잡한 산수연산부, 조종부 그리고 고수준프로그람언어와 콤퓨터**체계프로그람**들이 장비되였다.

제 2 세대콤퓨터에서는 또한 DEC 회사가 적지 않은 역할을 놀았다. DEC 는 1957 년에 나왔으며 그때 PDP-1 이라는 첫 콤퓨터를 출품하였다. 이 콤퓨터는 제 3 세대로의 돌파구를 여는 소형콤퓨터의 시작이였다.

IBM 7094

1952 년에 제작된 700 계렬로부터 1964 년에 제작된 7000 계렬의 마지막번호까지 IBM 제품계렬들은 콤퓨터제품의 표준적발전을 급속히 촉진하였다. 제품계렬의 련속적인 번호들은 성능증가, 용량증가, 저가격화를 보여 준다.

표 2-3 은 이 경향성을 설명해 준다. 주기억기의 크기는 1K × 36bit 단어의 배수로서 2k로부터 32k단어까지 증가하였으며 기억기의 한 단어접근시간 즉 **기억기주기시간**은 30 μs 로부터 1.4 μs로 단축되였다. 조작코드의 수는 최대 24 개로부터 185 개까지 늘어났다.

표 2-3의 마지막렬은 CPU 의 상대실행속도를 보여 준다. 속도의 증가는 전자회로의 갱신과 더 복잡한 회로를 구성함으로써 달성되였다. 실례로 IBM7094 는 다음명령을 완충하기 위한 명령보관등록기를 가진다. 조종부는 명령을 꺼낼 때에 기억기로부터 두개의

린접한 명령을 꺼낸다. 일반적으로 드문 분기명령의 발생을 제외하고 이 방법은 조종부가 명령꺼내기주기의 절반으로 한개 명령을 꺼내게 한다는것을 의미한다. 이 명령미리꺼내기는 평균명령주기시간을 단축한다.

표 2-3 의 나머지렬들은 본문에서 제시한 내용에 의해서 명백해 진다.

표 2-3. IBM700/7000 들의 실례

형번호	출현년도	CPU 기술	기억기기술	주기시간, μs	기억기용량 (k 단어)	조작코드의 수	첨수등록기의 수	류점수 연산기구	입출력통로	명령미리꺼내기	속도(701 에 대한 비)
701	1952	전자관	정전자관	30	2∼4	24	0	없다	없다	없다	1
704	1955	전자관	자심	12	4∼32	80	3	있다	없다	없다	2.5
709	1958	전자관	자심	12	32	140	3	있다	있다	없다	4
7090	1960	3 극소자	자심	2.18	32	169	3	있다	있다	없다	25
7094I	1962	3 극소자	자심	2	32	185	7	있다 (배정확도)	있다	있다	30
7094II	1964	3 극소자	자심	1.4	32	185	7	있다 (배정확도)	있다	있다	50

그림 2-5 에는 2 세대콤퓨터를 상징하는 IBM7094 의 구성을 보여 주었다. 보는바와 같이 IAS 콤퓨터와 별로 차이가 없다. 가장 기본적인 차이는 자료통로의 리용이다. 자료통로는 이 콤퓨터가 가지고 있는 처리장치와 명령모임과 독립적이라고 할수 있는 I/O 모

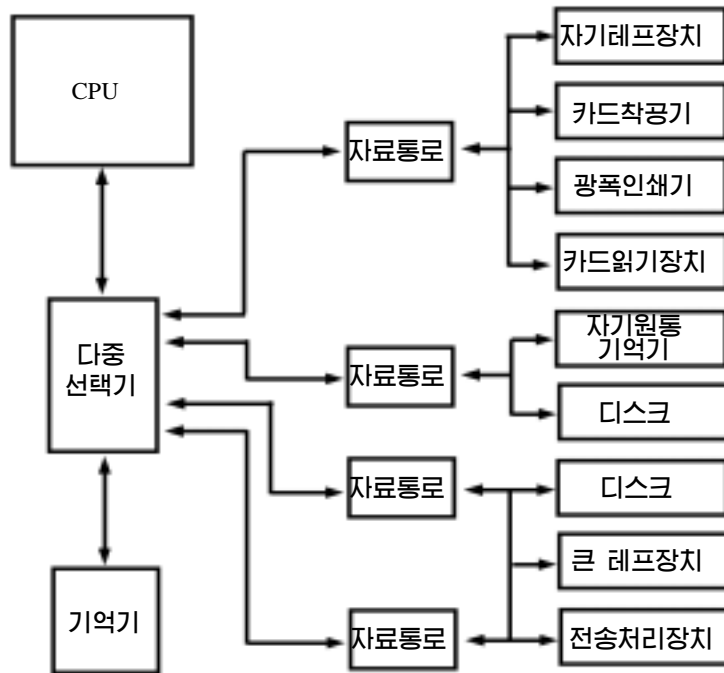



그림 2-5. IBM 7094 구성

들이다. 이와 같은 장치들로 구성된 콤퓨터체계는 CPU 가 세부적인 I/O 명령들을 실행하지 않는다. 세부적인 I/O 명령들은 주기억기에 기억되여 있으며 **자료통로**에 있는 전문적인 처리장치에 의하여 실행된다. CPU 는 자료통로에 조종신호를 보내고 기억기에 있는 명령순서배렬을 통로가 실행하도록 지시함으로써 I/O 전송에 착수한다. 자료통로는 연산을 진행하는 CPU 와 CPU 의 신호들과 독립적으로 이 과제를 수행한다. 이 구성은 처리부하가 많은 CPU 를 해방시켜 준다.

다른 새로운 특징은 자료통로, CPU, 기억기를 위한 중심마디점인 **다중분배기**이다. 다중분배기는 CPU 와 자료통로들이 독립적으로 동작하도록 하면서 기억기에 대한 접근을 진행하도록 순서화한다.

3. 제 3 세대: 집적회로콤퓨터

개별적인 3극소자로 이루어 진 구성부분을 **분리형구성부분**이라고 한다. 1950 년대와 1960 년대 초까지 전자장치들은 3극소자, 저항과 축전기 등으로 구성되였다. 분리형구성부분들은 따로따로 하나하나 제작되여 자기들의 용기에 포장되며 회로기판에 배선 혹은 납땜되여 콤퓨터나 오씰로그라프 등 전자제품들에 설치된다. 3 극소자로 이루어 졌다고 하는 전자장치 모두는 소자의 다리가 회로기판의 작은 구멍에 납땜되여 있다. 3극소자로부터 회로기판까지의 제작공정은 비용이 많이 들며 복잡하다.

이 사실들은 콤퓨터산업에서 문제로 되였다. 제2세대 콤퓨터들은 대략 10000개의 3극소자를 가지고 있었다. 이 구성은 몇십만개로 늘어 났으며 더 새롭고 더 능률적인 전자장치를 제작하기 어렵게 하였다.

1958 년에 전자혁명에 의하여 집적회로가 발명되여 미소전자시대가 시작되였다. 콤퓨터의 제 3 세대는 집적회로로 되였다. 이 절에서 집적회로기술을 간단히 소개한다. 그 다음 3 세대의 시작으로 되는 가장 표준적인 두개의 콤퓨터 IBM System/360 과 DEC PDP-8 을 고찰한다.

미소전자

미소전자는 글자그대로 《작은 전자》라는 의미이다. 수자회로와 콤퓨터산업의 시작으로부터 수자회로의 크기를 감소시키는것이 추세로 되였다. 이 추세의 함축된 의미와 우점을 설명하기전에 수자회로의 본질에 대하여 고찰하여야 한다. 상세한 설명은 부록 I 에서 찾으면 된다.

우리가 알고 있는 수자형콤퓨터의 기본요소들은 기억, 전송, 처리, 조종기능을 수행하여야 한다. 구성부분은 다만 두가지 기본형태로서 문소자들과 기억세포만을 요구한다 (그림 2-6). 문소자는 IF A AND B ARE TRUE THEN C IS TRUE(AND 문)와 같은 간단한 불 혹은 론리기능을 실행하는 소자이다. 이와 같은 소자들은 수문들이 하는것과 같은 방

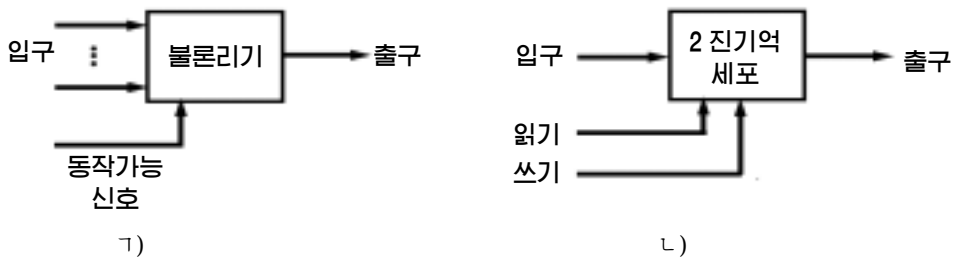



그림 2-6. 기초콤퓨터요소

ㄱ-문, ㄴ-기억기세포

법으로 자료흐름을 조종하기때문에 문소자라고 한다. 기억세포는 자료의 한개 비트를 기억시킬수 있는 요소이다. 즉 이 요소는 임의의 순간에 두개의 안정상태중의 한 상태로 될수 있다. 이 기본요소들을 많이 결합하여 콤퓨터를 구성한다. 우리는 다음과 같은 4 개의 기본기능들에 이것을 련관시킬수 있다.

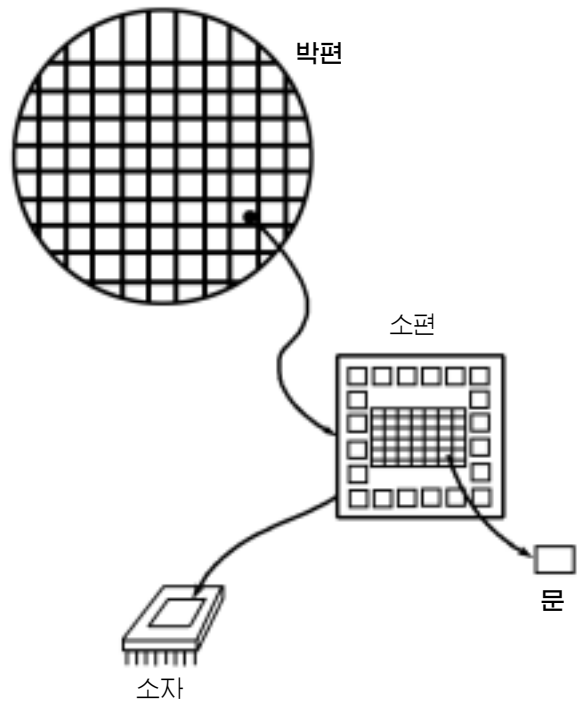
- **자료기억:** 기억세포들에 의하여 이루어 진다.
- **자료처리 :** 문소자들에 의하여 이루어 진다.
- **자료전송 :** 구성부분들사이의 경로는 기억기로부터 기억기에로 그리고 기억기로부터 문소자를 통하여 기억기에로 자료를 전송하는데 리용된다.
- **조종:** 구성부분들사이의 경로는 조종신호를 전송할수 있다. 실례로 문소자는 하나 혹은 두개의 자료입구들과 문을 능동상태로 만드는 조종신호입구를 가질수 있다. 조종신호가 ON 으로 될 때 문소자는 자료를 입력하고 출력한다. 마찬가지로 기억세포는 WRITE 조종신호가 ON 일 때 그 입구선의 비트를 기억하며 READ 조종신호가 ON 일 때 그의 출구선으로 비트를 출구한다.

결국 콤퓨터는 문과 기억세포들의 호상결합으로 구성된다. 또한 문과 기억요소들은 간단한 수자회로요소들로 구성된다.

집적회로는 3 극소자, 저항, 도체들과 같은 구성요소들이 규소와 같은 반도체로 만들어 진다는 원리를 리용하였다. 그것은 같은 회로에서 분리된 규소소편으로부터 만들어진 분리형구성부분들을 조립하는것이 아니라 매우 작은 규소소편에 전체 회로를 제작하는 고체상태기술의 확장이다. 많은 3 극소자들이 한개의 얇은 규소박편우에 동시에 제작된다. 여기서 중요한것은 이 3 극소자들이 도금처리에 의하여 련결되여 회로를 형성할수 있다는것이다.

그림 2-7 에 집적회로의 기본개념을 주었다. 얇은 규소박편은 매개가 수평방미리메터인 작은 령역들의 행렬로 분할된다. 이 매 작은 령역에 동일한 회로들이 제작되여 박편이 소편들로 된다. 매개 소자들은 많은 문과 기억세포들 그리고 여러개의 입출구련결점들로 이루어진다. 그다음 이 소편은 보호를 위하여 함침되며 다른 장치들과 련결하기 위한 단자들이 제공된다. 이 집적회로들이 더 크고 복잡한 회로를 구성하도록 인쇄기판에 서로 련결된다.

그림 2-7. 박편, 소편, 소자사이관

초기에는 적은 수의 문회로 혹은 기억세포들이 한개 소편에 견고하게 제작되였다. 초기에 이 집적회로들은 소규모집적회로(SSI)라고 불렀다. 날이 감에 따라 더 많은 구성부분들을 같은 소편에 묶을수 있게 되였다. 이 집적도의 장성을 그림 2-8 에 주었다. 즉 그림에서 보여 준것이 가장 주목할만한 기술적발전추세의 하나이다. 이 그림은 유명한 무

어의 법칙을 반영하며 이 법칙은 1965 년에 인텔의 공동개발자인 고든 무어(Gordon Moore)에 의하여 제안되였다[MOOR65]. 무어는 한 소편에 들어 갈수 있는 3 극소자의 수가 매해 2 배 늘어 나며 이러한 경향은 가까운 앞날에도 계속될것이라는것을 정확히 예측하였다. 더 놀라운것은 이 속도가 세월이 흐를수록 계속 유지되였다는것이다. 1970 년대 18 개월동안 이 속도가 떠졌으나 그후 이 속도는 유지되였다.

무어법칙의 결과는 다음과 같다.

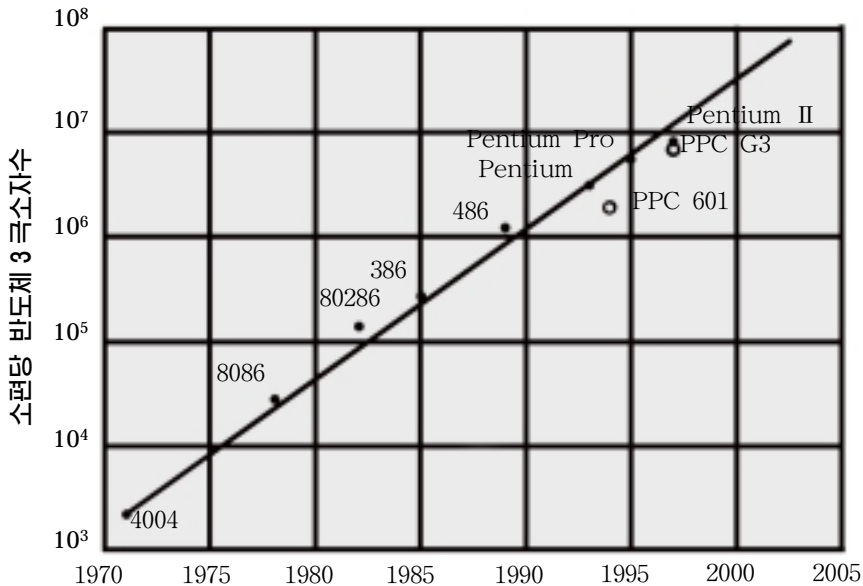



그림 2-8. CPU 3 극소자수의 증가특성

- 소편의 가격은 집적도가 급속히 장성하는 동안에 실제적으로 변하지 않고 유지되였다. 이것은 콤퓨터론리회로와 기억회로의 가격이 급격히 떨어 졌다는것을 의미한다.
- 론리회로와 기억요소들이 보다 조밀한 소편에 함께 배치되여 있으므로 배선의 길이가 짧아 지고 연산속도가 증가한다.
- 콤퓨터는 더 작아 지며 여러가지 환경들에 설치되여 더 쓰기 편리해 진다.
- 전력과 랭각요구가 감소된다.
- 집적회로에서 호상련결은 납땜에 의한 련결보다 더 믿음성이 있다. 매개 소편들이 더 많은 회로를 가지게 하면 소편들사이 련결은 보다 적어 질것이다.

IBM System/360

1964 년에 IBM 은 7000 계렬콤퓨터를 출품하여 콤퓨터시장에서 확고한 지위를 차지하였다. 그때 IBM 은 콤퓨터의 새로운 계렬인 system/360 을 개발하였다. 그자체의 개발은 놀랍지 않았지만 그때의 IBM 구입자들에게는 새롭게 일종의 불쾌감을 주었다. 즉 360 계렬은 지난 날의 IBM 기종들과 호환성이 없었다. 따라서 그때 사용자들을 360 에로 전환시키기 어려웠다. 이것은 IBM 에 의한 대담한 결심이였다. 그러나 IBM 이 생각하는것은 7000 계렬의 일부 제한성을 없애고 새로운 집적회로기술발전에 상응한 체계를 생산하여야 한다는것이다. 전략은 재정적으로나 기술적으로 수지가 맞았다. 360 은 10 년간 성과를

이룩하였고 70 %가 넘는 시장을 차지하여 콤퓨터사용자들을 우세하게 독점함으로써 IBM은 아주 견고해 졌다. 그리고 일부 변경되고 확장된 360방식은 오늘까지 IBM의 대형기방식으로 유지되였다. 이 방식을 리용한 실례들을 이 본문을 통하여 고찰할수 있다.

Systerm/360은 공업적으로 콤퓨터의 첫 계획화된 계렬이였다. IBM계렬에서 성능과 가격은 넓은 범위를 가진다. 표 2-4에 1965년의 여러가지 형에 대한 기본특성을 지적하였다(계렬의 매개 형은 형번호에 의하여 구별된다.). 형들은 실행시간에서 차이나지만 계렬의 낮은급형에서 작성된 프로그람은 계렬의 높은급형에서 실행할수 있다.

표 2-4. System/360 계렬의 기본특성

특성	30형	40형	50형	65형	75형
최대기억용량, byte	64K	256K	256K	512K	512K
기억기자료속도, Mbyte/s	0.5	0.8	2.0	8.0	16.0
처리장치 주기시간, μs	1.0	0.625	0.5	0.25	0.2
상대속도	1	3.5	10	21	50
자료통로의 최대수	3	3	4	6	6
한 통로의 최대자료속도, kbyte/s	250	400	800	1250	1250

호환성콤퓨터들의 계렬화개념은 새롭고 성공적이였다. 적당한 요구와 적은 예산을 요구하는 구입자들은 상대적으로 눅은 30형으로부터 시작하였다. 후에 구입자들의 요구가 늘어 나면 이미 개발된 프로그람에 대한 투자를 잃어 버리지 않고도 더 빠른 콤퓨터와 더 많은 기억기로 갱신할수 있다. 계렬화의 특징은 다음과 같다.

- **류사한 혹은 동일한 명령모임** : 많은 경우 꼭 같은 기계어명령모임이 계렬의 모든 형들에서 보장된다. 따라서 어느 한 콤퓨터에서 실행되는 프로그람은 다른 임의의 콤퓨터에서도 실행될수 있다. 일부 경우 계렬의 제일 낮은 급의 콤퓨터는 제일 높은 급의 부분적인 명령모임을 가진다. 이것은 프로그람들이 높은 급에서는 실행될수 있으나 높은 급에서 개발된 프로그람들이 낮은 급에서는 실행될수 없다는것을 의미한다.
- **류사한 혹은 동일한 조작체계** : 같은 기본조작체계는 모든 계렬의 형들에서 리용될수 있다. 일부 경우 추가적인 특성들은 더 높은급 콤퓨터들에 추가된다.
- **속도증가** : 명령실행속도는 낮은 급으로부터 높은 급으로 올라 가면서 증가한다.
- **I/O포구수의 증가** : 낮은 급으로부터 높은 급으로 가면서 증가한다.
- **기억용량의 증가** : 같은 계렬의 낮은 급으로부터 높은 급으로 올라 가면서 증가한다.
- **가격증가** : 같은 계렬의 낮은 급으로부터 높은 급으로 올라 가면서 증가한다.

이와 같은 계렬화개념은 어떻게 실현되는가? 차이점들은 3가지 요인 즉 속도, 크기, 동시성정도에 기초한다. 실례로 주어 진 명령의 실행에서 더 높은 속도는 병렬로 출구하는 보조연산들을 가지는 ALU의 더 복잡한 회로들을 리용함으로써 달성된다. 속도증가의 또 하나의 다른 방법은 주기억기와 CPU 사이의 자료통로의 폭을 넓히는것이다. 30형에서는 한번에 주기억기로부터 1byte의 꺼내기를 할수 있으며 반면에 70형에서는 한번에 8byte를 꺼낼수 있다.

system/360은 IBM의 앞으로의 방향을 지적하지 않았지만 전반적산업에서 크게 영향을 주었다. 이러한 대부분의 특성들이 다른 대형콤퓨터들에서 표준화되였다.

DEC PDP-8

IBM 이 처음으로 system/360 을 운영하던 그때 또 하나의 DEC 의 PCP-8 이라는 콤퓨터가 개발되였다. 보통 콤퓨터가 공기조화기가 달린 방을 요구할 때 PDP-8 은 실험실 걸상우에 올려 놓거나 다른 기구에 설치할수 있을 정도로 작았다. PDP-8 은 대형콤퓨터가 할수 있는 모든 일을 할수 없었다. 그러나 16,000 딸라로서 콤퓨터를 구입하려는 실험실전문가들에게는 충분히 눅은 가격이였다. 대비적으로 대형콤퓨터인 system/360 계렬은 PDP-8 개발의 몇달전에 수십만딸라의 가격으로 소개되였다.

표 2-5. PDP-8 의 발전[VOEL88]

형	첫 상품화 번호	처리장치+4K 12bit 단어기억기의 가격 (천딸라)	기억기의 자료속도 (단어/μs)	체적 (립방피트)	혁신과 개선
PDP-8	4/65	16.2	1.26	8.0	자동배선도입
PDP-8/5	9/66	8.79	0.08	3.2	명령계렬화실현
PDP-8/1	4/68	11.6	1.34	8.0	중규모집적회로
PDP-8/L	11/68	7.0	1.26	2.0	소규모캐비네트
PDP-8/E	3/71	4.99	1.52	2.2	전용모선
PDP-8/M	6/72	3.69	1.52	1.8	8/E 의 절반정도의 캐비네트
PDP-8/A	1/75	2.6	1.34	1.2	반도체기억기 류점수처리장치

PDP-8 의 눅은 가격과 작은 크기는 PDP-8 을 사들여 재판매를 실현하려는 또 다른 제작자들을 만들었다. 이 제작자들을 기본기구제작자들(OEMs)이라고 알려 져 있으며 OEM 시장은 콤퓨터시장의 기본부분을 이루고 유지하였다.

PDP-8 은 즉시 인기를 끌었으며 DEC 회사가 부흥하게 하였다. 이 콤퓨터와 이것을 계승한 PDP-8 계렬의 다음형(표 2-5 를 참고)들은 IBM 콤퓨터들에서 이전에 보존하였던 생산지위를 차지하였으며 5 만딸라로 12 년동안 팔렸다. DEC 의 공식적인 력사가 시작되여 PDP-8 은 수십억딸라산업을 위한 방향으로 유도하면서 소형콤퓨터들의 개념을 세웠다. 또한 DEC는 첫 소형콤퓨터회사로서 설립되였으며 그 시기에 PDP-8이 절정에 이르러 DEC 는 IBM 다음가는 콤퓨터제작자로 되였다.

IBM 이 700/7000 과 system/360 체계들에서 리용한 중심절환방식에 비하여 PDP-8 후에 제작된 형들은 실제적으로 소형콤퓨터와 극소형콤퓨터들에 공동으로 쓰이는 모선구조를 리용하였다. 이것을 그림 2-9 에서 설명하였다. 전용모선이라고 하는 PDP-8 모선은

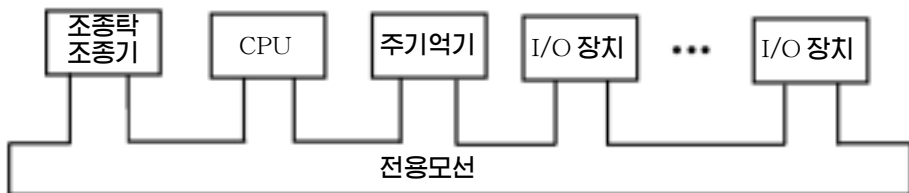



그림 2-9. PDP-8 모선구조

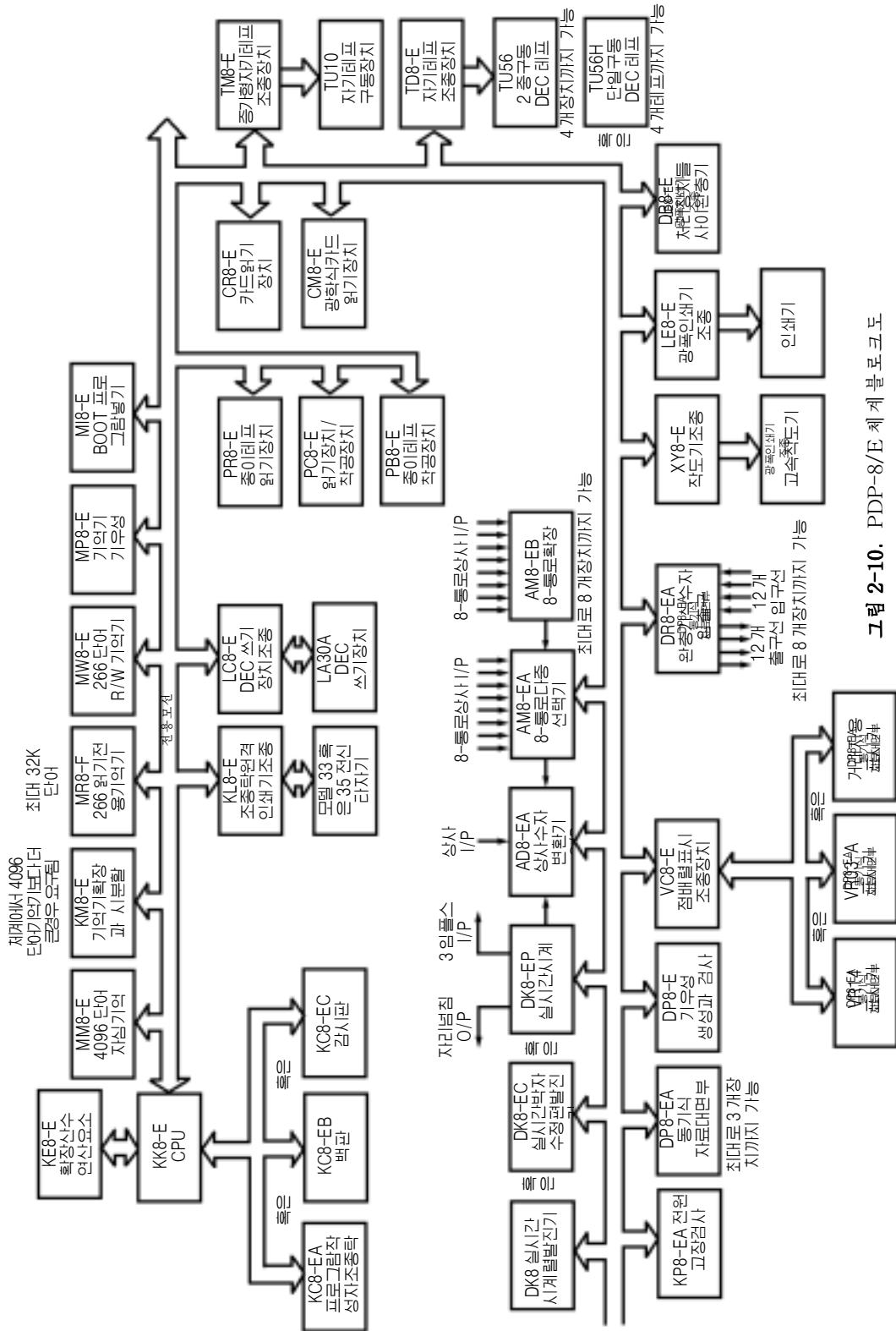



그림 2-10. PDP-8/E 체계블로크도

조종, 주소, 자료신호들을 전송하기 위한 96 개의 개별적인 신호선들로 구성되였다. 모든 체계구성부분들은 신호경로들의 공동모임을 공유하기때문에 그것들의 리용은 CPU 에 의하여 조종되여야 한다. 이 방식은 아주 유연하며 여러가지 구성들을 만들기 위하여 모선에 장치들을 결합하는것을 허락한다. 그림 2-10 에 PDP-8/E 구성을 보여 주었다.

4. 다음세대의 콤퓨터

3 세대를 지나 콤퓨터의 세대를 정의하는 경우 일반적으로 의견이 일치되지 않고 있다. 표 2-2 에서는 집적회로기술의 발전에 기초하여 4 세대와 5 세대를 분류하였다. 한개의 집적회로소편에 1,000 개이상의 구성요소가 배치되여 있는 대규모집적회로(LSI)와 소편당 구성요소수가 10,000 개이상인 초대규모집적회로(VLSI)가 있으며 현재 VLSI 소편은 100,000 개의 구성요소들을 가지고 있다.

기술의 빠른 속도, 새 제품들의 빠른 출현 그리고 장치와 같이 프로그람과 통신의 중요성으로 하여 세대의 구별은 명백치 않다. 새로운 발전들의 상품적응용은 이미 1970 년대에 기본변화를 가져 왔으며 아직까지 이 변화들이 계속되고 있다고 말할수 있다. 이 절에서는 가장 중요한 두가지 문제에 대하여 고찰한다.

반도체기억기

콤퓨터를 위한 반도체집적회로기술의 첫 응용은 집적회로소편들에서의 처리장치(조종단위와 산수 및 론리단위)의 실현이였다. 또한 이와 같은 기술은 기억기구성에도 리용되였다.

1950 년대와 1960 년대에 거의 모든 콤퓨터기억기는 자심기억기였으며 자심기억기의 직경은 대략 1/16 인치였다. 자심들은 콤퓨터안의 작은 쇠그물형태의 도선들에 매달려 있었다. **자심**의 자화된 한 방향은 1 로 표현하며 자화된 다른 방향은 0 으로 표현한다. 자심기억기는 속도가 비교적 빠르다. 즉 기억기에 기억된 1bit 읽기는 초당 백만회로서 짧은시간이 걸린다. 그러나 이것은 비용이 많이 들고 체적이 크며 읽을 때에 자료를 잃어 버린다. 즉 자심에 대한 읽기를 하면 거기에 기억된 자료는 지워 진다. 그러므로 반드시 자료를 읽자마자 즉시로 자료를 재기억시키는 회로가 있어야 한다.

그후 1970 년에 패어챠일드는 처음으로 용량이 큰 반도체기억기를 만들었다. 이 소자는 한개의 자심기억기알갱이만한 크기를 가지며 256bit 의 기억용량을 가지였다. 자심기억기에 비하여 속도가 더 빠르고 자료를 잃어 버리지 않았다. 비트읽기속도는 초당 700 억회였다. 그러나 자심기억기보다 비트당 가격이 더 높았다.

1974 년에 근본적인 변화가 일어 났다. 즉 반도체기억기의 비트당 가격이 자심기억기의 비트당 가격보다 떨어 졌다. 물리적인 기억기밀도의 증가에 따르는 기억기가격은 련속적으로 그리고 빠르게 떨어 졌다. 이것은 더 작고 더 빠르며 더 큰 기억기용량을 가진 콤퓨터방향으로 몇년동안에 그 값도 서서히 비싸 졌다. 기억기기술에서의 발전은 다음에 론의하게 되는 처리장치기술의 발전과 함께 10 년 남짓한 기간에 콤퓨터의 본질을 변화시켰다. 비록 체적이 크고 값 비싼 콤퓨터들이 관상용으로 남아 있지만 콤퓨터는 사무용콤퓨터와 개인용콤퓨터와 같은 사용자용으로 변화되였다.

1970 년부터 반도체기억기는 10 개의 세대를 거쳐 즉 한 소편에 1kbit, 4kbit, 16kbit, 64kbit, 256Kbit, 1Mbit, 4Mbit, 16Mbit, 64Mbit, 256Mbit 를 기억시킬수 있는 단계들로 발전하였다. 매개 세대는 전 세대에 비하여 4 배의 기억밀도를 가지며 접근속도가 빨라 지고 비트당 가격이 떨어 졌다.

극소형처리장치

기억기소편들에서 요소들의 밀도가 계속 올라 감에 따라 처리장치소편들에서 요소들의 밀도도 계속 올라 간다. 시간이 갈수록 더 많은 요소들이 한 소편에 집적화되며 따라서 한개의 콤퓨터처리장치를 구성하는데 더 적은 소편들이 요구된다.

인텔이 4004 를 개발하였을 때인 1971 년에 돌파구가 열리였다. 4004 는 한개 소편에 CPU 의 구성부분모두를 포함시키는 첫 소편이였으며 이때 극소형처리장치가 나오게 되였다.

4004 는 두개의 4bit 수를 더하기할수 있으며 더하기반복방법으로 곱하기연산을 할수 있다. 4004 는 발전성이 없는 초기제품이였지만 극소형처리장치능력의 련속적인 진화의 전망을 열어 놓았다.

이 진화는 처리장치가 한번에 처리하는 비트의 수에 의하여 아주 쉽게 표현할수 있다. 이것을 명백히 밝히는 척도는 없지만 대체로 처리장치가 한번에 입력하거나 출력할수 있는 자료의 비트수인 자료모선폭이 가장 좋은 척도라고 생각된다. 다른 하나는 축적기 혹은 일반등록기들에서의 비트수이다. 흔히 이 비트수들이 일치하지만 항상 그렇지는 않다. 실례로 극소형처리장치들의 일부는 등록기들에서 16bit 수로 연산하게 개발되였지만 한번에 8bit 씩 읽기쓰기할수 있다.

극소형처리장치의 진화에서 다음의 기본단계는 1972 년에 개발된 Intel 의 8008 이다. 이것은 처음으로 되는 8bit 극소형처리장치였으며 4004 를 2 개 복합시켰다. 이 단계들은 1974 년에 Intel 이 8080 을 개발하는데 아무런 영향도 미치지 못하였다. 8080 은 처음으로 되는 일반용극소형처리장치였다. 4004 와 8008 은 특수한 응용을 위하여 설계되였다. 8080 은 일반용극소형콤퓨터를 위한 CPU 로 설계되였다. 8008 과 같이 8080 은 8bit 극소형처리장치이다. 그러나 8080 은 더 빠르고 더 풍부한 명령모임을 가지며 더 큰 주소화능력을 가진다. 거의 같은 시기에 16bit 극소형처리장치들이 개발되기 시작하였다. 그러나 위력한 일반용극소형처리장치들이 개발될 때인 1970 년대 말까지는 쓰이지 않았다. 그것들중 하나가 8086 이였다. 다음단계는 벨라브와 휼레트 팩카드가 32bit 한소편극소형처리장치를 개발할 때인 1981 년이였다. 인텔은 1985 년에 32bit 극소형처리장치 80386 을 개발하였다(표 2-6).

표 2-6. 인텔극소형처리소자의 발전

ㄱ)1970 년대 극소형처리장치들

	4004	8008	8080	8086	8088
생산날자	11/5/71	4/1/72	4/1/74	6/8/78	6/1/79
박자속도	108KHz	108KHz	2MHz	5MHz, 8MHz, 10MHz	5MHz, 8MHz
모선폭	4bit	8bit	8bit	16bit	8bit
반도체수(미크론)	2,300(10)	3,500	6,000(6)	29,000(3)	29,000(3)
주소가능한 기억기	640byte	16kbyte	64kbyte	1Mbyte	1Mbyte
가상기억기	—	—	—	—	—

ㄴ)1980 년대 극소형처리장치들

	80286	인텔 386TM DX 처리장치	인텔 386TM SX 처리장치	인텔 486TM DX CPU 처리장치
생산날자	2/1/82	10/17/85	6/16/88	4/10/89
박자속도	6MHz-12.5MHz	16MHz-33MHz	16MHz-33MHz	25MHz-50MHz
모선폭	16bit	32bit	16bit	32bit
반도체수(미크론)	134,000(1.5)	275,000(1)	275,000(1)	1.200000(0.8-1)
주소가능한 기억기	16Mbyte	4Gbyte	4Gbyte	4Gbyte
가상기억기	1Gbyte	64Tbyte	64Tbyte	64Tbyte

ㄷ)1990 년대 극소형처리장치들

	인텔 486TM SX 처리장치	Pentium 처리장치	Pentium Pro 처리장치	Pentium II 처리장치
도입날자	4/22/91	3/22/93	11/01/95	5/07/97
박자속도	16MHz-33MHz	60MHz-166MHz	150MHz-200MHz	200MHz-300MHz
모선폭	32bit	32bit	64bit	64bit
반도체수(미크론)	1185000 (1)	3100000(8)	5500000(0.6)	7500000
주소가능한 기억기	4Mbyte	4Gbyte	64Gbyte	64Gbyte
가상기억기	64Gbyte	64Tbyte	64Tbyte	64Tbyte

제 2 절. 성능설계

해마다 콤퓨터체계의 가격은 급격히 떨어 지는 한편 콤퓨터체계의 성능과 능력은 급격히 올라 간다. 지방의 콤퓨터상점들에서도 10 년전의 IBM 대형콤퓨터의 위력과 맞먹는 1000 딸라이하의 개인용콤퓨터들을 살수 있다. 극소형처리장치, 기억기와 다른 소자들을 포함하는 개인용콤퓨터안에는 1 억개정도의 3 극소자가 있다. 그만한 돈으로 임의의 물건을 1 억개 살수는 없다. 휴지종이라고 해도 10 만딸라이상 들것이다.

이처럼 위력한 콤퓨터를 아주 렴가로 살수 있게 되였다. 그리고 계속적인 기술발전으로 하여 몹시 놀랄만큼 복잡하고 위력한 응용프로그람을 개발할수 있게 되였다. 실례로 오늘 극소형처리장치에 기초한 위력한 체계들을 요구하는 탁상형콤퓨터응용프로그람들은 다음과 같다.

- 화상처리프로그람
- 음성인식프로그람
- 비데오회의프로그람
- 다매체저작프로그람
- 파일들의 음성 및 영상해설프로그람

현재 워크스테이션체계들은 모형화체계들과 같은 더 높은 급의 공학과 파학응용프로그람들을 지원하며 화상과 비데오응용프로그람들로 작업그룹원리들을 적용하기 위한 능력을 지원한다. 더우기 봉사는 업무와 자료기지처리를 조종하고 지난기간의 큰 대형콤퓨터집중처리체계를 대신하는 방대한 의뢰기-봉사기망들을 지원하기 위한 효률적인 봉사들에 점차적으로 의거하고 있다.

콤퓨터구성과 구성방식의 관점으로부터 이 모든것들을 고찰하면 하나는 현대콤퓨터들이 50 년전의 IAS 콤퓨터와 본질적으로 같다는것이며 다른 하나는 가까운 앞날에 현대적인 콤퓨터들의 성능을 최대로 달성할수 있는 기술이 점차적으로 준비되고 있다는것이다.

이 책에서 콤퓨터의 세부요소와 구성부분들을 고찰할 때 두가지 대상에 주목하게 된다. 첫째로는 이 책이 고찰하는 모든 범위에서 기본적인 기능들을 설명하는것이고 둘째로는 이 책이 최대성능을 달성하기 위하여 요구되는 기술들을 주는것이다. 이 절의 나머지부분들에서 성능설계에 필요한 일부 구동요소들을 명백히 서술하였다.

1. 극소형처리장치의 속도

놀랄만한 위력을 가진 Pentium이나 PowerPC를 소개한것은 처리장치소편제작자들에 의하여 처리장치의 속도가 높아 졌기때문이다. 이 처리장치들의 속도는 이미전에 말한바와 같이 무어의 법칙에 따라 발전한다. 이 법칙이 성립하는 기간 처리장치소편제작자들은 매 3년마다 3극소자의 집적도를 4배로 증가시킨 새로운 소편을 만들것이다. 기억기소편에서도 여전히 3년주기로 콤퓨터주기억기의 기본기술인 동적자유접근기억기(DRAM)의 용량이 4배로 높아 진다. 극소형처리장치들에서 새로운 회로가 추가되고 요소들사이의 거리를 축소한데로부터 오는 속도증가로 인텔도 1978년에 x86계렬을 내놓은 때로부터 3년주기로 성능을 4~5배로 증가시켰다.

그러나 극소형처리장치의 본래속도는 콤퓨터명령형식에 따라 진행하는 작업의 정상적인 흐름을 보장하지 않고서는 자기의 잠재적속도에 다 도달할수 없다. 이 순조로운 흐름을 얻기 위한 그 어떠한 방법들도 처리소자의 능력에는 도달하지 못한다. 따라서 소편제작자들이 집적도가 더욱 높은 소편을 어떻게 제작할것인가하는 연구사업을 진행하는 한편 처리소자설계가들은 이 불균형을 해결하기 위하여 더 세련된 기술들을 개발하여야 한다. 현대처리소자들을 구축하는 기술들가운데는 다음과 같은것이 있다.

- **분기예측**: 처리장치들은 먼저 실행한 프로그람의 앞부분에 대한 고찰을 진행하여 명령들의 분기점을 예측하고 다음처리를 순조롭게 한다. 만일 처리장치가 거의 모두를 추측하였다면 정확한 명령미리꺼내기를 진행하고 처리장치가 정상적으로 연산하도록 그것을 완충할것이다. 이 방법의 더 복잡한 실례는 다음의 분기가 명백하지 않은것을 예측하는것 즉 분기선두가 여러개인것이다. 따라서 분기예측은 더 복잡해 진다.
- **자료흐름분석**: 처리장치는 최적인 명령순서를 만들기 위하여 어떤 명령들이 다른 결과 혹은 자료에 의존되는가를 분석한다. 실제로 기본프로그람의 순서와 관계없이 명령들은 준비되면 실행될수 있게 순서화된다. 이것은 불필요한 지연을 막는다.
- **투기적실행**: 분기예측과 자료흐름분석을 리용하면서 일부 처리장치들은 프로그람실행에 선행하여 일부 명령들을 먼저 투기적으로 실행하며 그 결과들을 림시적인 위치에 보관한다. 이것은 처리장치로 하여금 필요한 명령들을 실행함으로써 가능한껏 실행기구들이 쉬임없이 동작하도록 한다.

이 기술들과 다른 복잡한 기술들은 처리장치의 능력을 완전히 발휘하도록 하는데 반드시 필요하다.

2. 성능의 균형화

처리장치능력이 매우 빠른 속도로 높아 지고 있지만 콤퓨터의 구성부분들은 그렇지 못하다. 따라서 성능의 균형화 즉 여러 구성부분들의 능력들가운데서 불균형을 보상하기 위하여 구성과 구성방식을 조정할것을 요구한다.

처리장치와 주기억기사이의 대면부에서 보다 더 심한 불균형들에 의하여 제기되는 문제들이 있다. 그림 2-11 에 그려 진 력사를 고찰하자. 처리장치의 속도와 주기억기의 용량이 빠른 속도로 증가하는 한편 자료가 주기억기와 처리장치사이에 전송될수 있는 속도는 대단히 느리다. 처리장치와 주기억기사이의 대면부는 기억기소편들과 처리장치소편사이에 프로그람명령들과 자료들의 정상적인 흐름을 보장하도록 되여야 하므로 전반적인

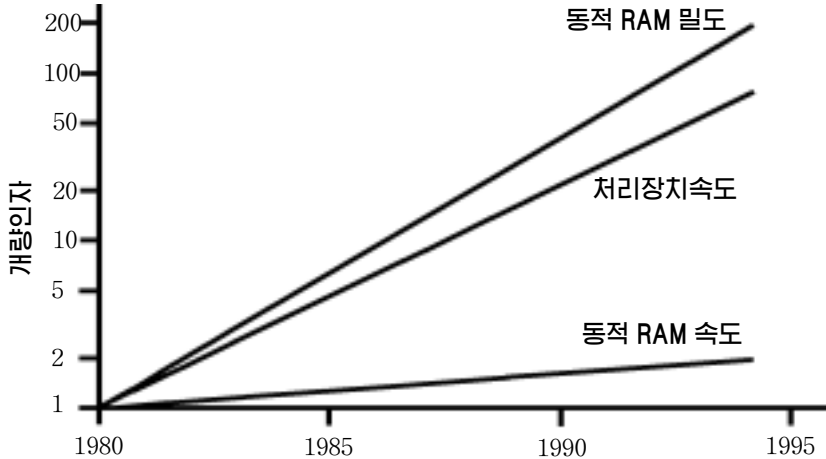



그림 2-11. DRAM 과 처리장치특성들의 발전

콤퓨터에서 가장 결정적인것은 전송통로이다. 만일 기억기와 전송통로가 처리장치의 요구를 따르지 못하면 처리장치는 기다림상태의 무효공간을 가지게 되며 귀중한 처리시간을 잃어 버리게 된다.

이 추세들의 영향을 그림 2-12 에서 생동하게 보여 주었다. 요구되는 주기억기의 총용량이 올라 가고 또한 DRAM 집적도도 빠른 속도로 올라 간다. 결과로서 평균적인 체계당 DRAM 수는 줄어 든다. 그림에서 굵은 검은선은 이 관계를 보여 주며 고정된 크기의 기억기에 대하여 요구되는 DRAM 의 수는 감소한다. 그러나 이것은 전송속도에 영향을 미친다. 그림에서 그늘진 부분은 체계의 특징들중 주기억기의 크기가 천천히 증가하면 DRAM 의 수가 감소한다는것을 보여 준다.

체계설계가들이 이 문제를 해결할수 있는 여러가지 방법들이 있는데 그것들 모두가 현재 콤퓨터설계들에 반영되였다. 실례로 다음과 같은것들을 들수 있다.

- 넓은 모선폭을 가진 DRAM 들을 제작하여 넓은 모선자료통로를 리용함으로써 한번에 전송하는 비트수를 증가시킨다.
- DRAM 소편에 캐쉬와 다른 완충용토막을 넣음으로써 더 능률적인 전송을 위한 DRAM 대면부를 변화시킨다.
- 처리장치와 주기억기사이에 점차 복잡하고 능률적인 캐쉬구조들을 삽입하여 기억기접근회수를 감소시킨다. 이것은 처리장치소편안에 있는 무소편캐쉬와 같이 처리장치소편에 하나이상의 캐쉬를 삽입시키는것도 포함한다.
- 고속모선, 완충기를 위한 모선들의 계층화 및 구조적인 자료흐름을 리용함으로써 기억기와 처리장치사이의 결합대역폭을 증가시킨다.

설계의 조건으로 되는 또 하나의 부분은 I/O 장치의 조종이다. 콤퓨터가 더 빠르고 더 높은 능력을 가질 때 집중적인 I/O 요구들로 주변장치의 리용을 지원하는것은 더 복잡한 응용프로그람들을 개발하게 한다. 표 2-7 에 개인용콤퓨터와 워크스테이션들에서 리용하는 전형적인 주변장치들의 실례를 보여 주었다. 이 장치들은 굉장한 자료처리능력을 요구한다. 현 세대처리장치들은 이 장치들에 의하여 얻어 진 자료들을 조종할수 있으며 처리장치와 주변장치사이에서 전송되는 자료를 받아 들이는 문제가 남는다. 여기서

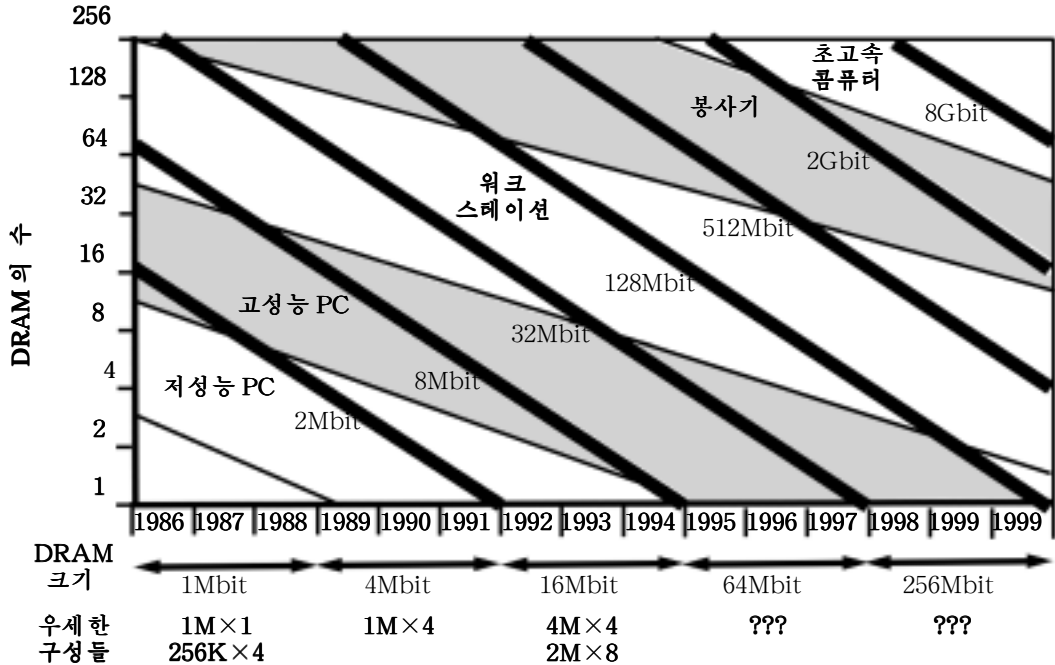



그림 2-12. DRAM 리용추세 [PRZY94]

이 전략들은 고속결합모선들과 모선들의 더 복잡한 구조를 리용하게 되는 고속완충기를 포함한다. 추가로 다중처리장치구성들을 리용하면 만족하는 I/O 요구를 도와 줄수 있다.

표 2-7. 여러가지 주변장치기술의 대표적인 대역폭요구

주변장치	기술	요구되는 대역폭, Mbyte
도형표시장치	24bit 색	30
국부망	100BASEX 혹은 FDDI	12
디스크조종기	SCSI 혹은 P1394	10
비데오	1024 × 768@30fps	67+
I/O 주변장치	기타	5+

이 모든것들의 기본은 균형을 보장하는것이다. 설계가들은 처리장치구성요소인 주기억기, I/O 장치 그리고 호상결합구조들의 처리능력과 처리들의 균형을 보장하기 위하여 끊임없이 노력한다. 이 설계는 끊임없이 발전하는 다음과 같은 두가지 요인들에 대처하기 위하여 연구되여야 한다.

- 여러가지 기술분야(처리장치, 모선들, 기억기, 주변장치들)들에서의 성능의 변화속도는 요소들의 형태에 따라 각이하다.
- 전형적인 명령형식과 자료접근방식에서 체계에 대한 요구의 내용들이 새로운 응용프로그람들과 새로운 주변장치들이 나옴에 따라 부단히 변한다.

따라서 콤퓨터설계는 방법론들을 계속 발전시키고 있다. 이 책에서는 이 방법론의 기초와 현 상태를 주었다.

제3절. Pentium과 PowerPC의 발전

이 책에는 콤퓨터의 개념과 대량생산을 설명하기 위하여 콤퓨터설계와 실현에 대한 구체적인 실례들을 많이 주었다. 이 책에서 취급하는 대부분의 실례는 두개의 콤퓨터계렬 즉 Intel의 Pentium과 PowerPC이다. Pentium에서는 복합명령모임콤퓨터(CISC)들에서 10년동안 설계노력의 결과를 고찰한다. CISC는 그 설계의 우수한 실례들로서 대형콤퓨터들과 슈퍼콤퓨터들, 봉사기들의 정교한 설계원리들을 통합하고 있다. PowerPC는 첫 RISC체계인 IBM801의 직접적인 계승이며 가장 위력하고 가장 설계가 잘된 RISC에 기초한 체계로서 인기가 높다.

이 절에서는 두 체계를 간단히 고찰한다.

1. Pentium

인텔은 10여년동안 극소형처리장치들을 만들어 온 제일가는 제작회사였다. 인텔이 처음으로 극소형처리소자제품을 개발함으로써 콤퓨터기술발전의 훌륭한 개척자로 되였다.

표 2-6은 그 발전을 보여 주고 있다. 흥미 있는것은 극소형처리장치가 빨리 발전할수록 더욱더 복잡해 진다는것이다. Intel은 실제적으로 가장 높은 수준에 올라 섰다. 인텔은 매 4년마다 새로운 형의 극소형처리장치들을 개발하군 하였다. 그러나 Intel은 개발기간을 1~2년으로 정하여 경쟁자들을 이기려고 하였으며 3개의 최신형펜티움계렬들을 내놓았다.

Intel제품계렬의 발전과정은 다음과 같다.

- **8080:** 세계에서 첫 일반목적처리장치이다. 8080은 8bit 처리장치이다. 8080은 첫 개인용콤퓨터 Altair에 리용되였다.
- **8086:** 더 능률이 높은 16bit 처리장치이다. 적은 량의 명령들을 실행하기전에 미리꺼내기하여 놓을수 있는 명령캐쉬 즉 기다림렬을 장비하였다. 이 처리소자의 변종인 8088은 Intel이 확고히 성공할수 있게 한 IBM의 첫 개인용콤퓨터에 리용되였다.
- **80286:** 8086의 확장으로서 1Mbyte대신에 16Mbyte주소화가 가능하다.
- **80386:** Intel의 첫 32bit 처리장치이며 기본적으로 높은 수준에서 검사된 제품이였다. 32bit구성방식으로서 80386은 이미 몇년전에 소개된 소형콤퓨터와 대형콤퓨터의 복잡성과 능력에 대한 경쟁을 하였다. 80386은 여러개의 프로그람을 동시에 실행하는 다중과제처리를 지원하는 첫 Intel처리장치였다.
- **80486:** 80486은 더 고급하고 정교한 캐쉬기술과 고급한 명령판호름조종의 리용을 도입하였다. 또한 기본 CPU로부터 복잡한 산수연산을 다른데로 넘겨 주지 않고 실행하도록 수값연산협동처리장치를 장비하였다.
- **Pentium:** Pentium에서 Intel은 병렬로 여러개의 명령을 실행하게 하는 고속스칼라기술을 리용하였다.
- **Pentium Pro:** Pentium Pro는 Pentium으로부터 시작된 고속스칼라구성에로의 이행을 계승하였으며 또한 등록기이름바꾸기, 분기예측, 자료흐름분석과 투기적인 실행을 추가하였다.

- **Pentium Ⅱ:** Pentium Ⅱ는 비데오, 음성과 도형자료들을 능률적으로 처리하도록 특수하게 설계된 Intel의 MMX 기술을 추가하였다.
- **Pentium Ⅲ:** Pentium Ⅲ과 3차원도형처리프로그람을 지원하는 추가적인 류점수명령들을 첨부하였다.
- **Merced:** Intel 처리장치의 이 새로운 세대는 64bit 연산을 진행하게 하였다.

2. PowerPC

1975년에 IBM에서 계획한 801소형콤퓨터는 RISC체계에서 리용되는 많은 구성방식의 개념들을 개척하였다. 801은 Berkeley RISC I 처리장치와 함께 RISC 구상을 내놓았다. 그러나 801은 설계구상을 립증하는 정도의 간단한 모형이였다. 1986년에 개발한 RTPC 실제제품에서는 801방식의 구상들을 실현하였다. RT PC는 상품화되지 못하였으나 비교할수 있을 정도로 더 좋은 성능을 가지였다. 1990년에 IBM은 801과 RT PC의 교훈으로부터 구축된 세번째 체계를 생산하였다. IBM RISC system/6000은 조금후에 개발된 고성능워크스테이션으로서 판매된 좋은 고속스칼라처리소자인 RISC였다. 이것을 IBM POWER 구성방식으로 부르기 시작하였다.

다음단계를 위하여 IBM은 68000 계렬의 극소형처리장치들을 개발한 Motorola와 Macintosh 콤퓨터에 Motorola 소편을 리용한 Apple 콤퓨터를 결합시켰다. 그 결과가 POWERPC를 실현한 처리장치계렬이다. 이 구성방식은 POWER 구성방식으로부터 유래되였다. 변화시킨것은 일부 명령들을 없애고 일부 까다로운 경우들을 없애기 위한 기술적요구들을 약화시킴으로써 기본적으로 있어야 할 기능들을 추가하고 더 효률적으로 실현할수 있도록 하였다. 결과 Power PC 방식은 RISC 체계이다. Power PC는 수백만대의 Apple Macintosh 콤퓨터와 많은 소편응용에 리용된다. 후자의 실례는 다중콤퓨터 관련회사의 가동환경을 가진 사용자들에게 일반관리접근을 제공하는 망설비에 꽂을수 있는 IBM 계렬의 망관리소편이다.

아래에는 PowerPC 계렬의 기본형들을 주었다(표 2-8).

표 2-8. PowerPC 처리장치개괄

	601	601/603e	604/604e	740/750(G3)	G4
첫 생산년도	1993	1994	1994	1997	1999
주기속도(MHz)	50-120	100-300	166-350	200-366	500
캐쉬	—	16kbyte 명령 16kbyte 자료	32kbyte 명령 32kbyte 자료	32kbyte 명령 32kbyte 자료	32kbyte 명령 32kbyte 자료
L2 2차 캐쉬 지원	—	—	—	256kbyte-1M byte	256kbyte-1M byte
3극소자수 (10^6)	2.8	1.6-2.6	3.6-5.1	6.35	

- **601:** 601의 목적은 될수록 빨리 시장에 PowerPC 구성방식을 보급하는것이다. 601은 32bit 처리장치이다.
- **603:** 낮은 탁상형과 휴대형콤퓨터로서 역시 32bit 처리장치이며 601과 성능에서

비슷하다. 그러나 더 가격이 눅으며 더 능률적이다.

- **604:** 탁상형과 말단봉사기를 위한것이다. 이것은 32bit 콤퓨터이며 더 높은 성능을 가지도록 하기 위하여 더 현대적인 고속스칼라설계기술을 리용하였다.
- **620:** 높은급봉사기용이다. 64bit 자료통로를 가지는 완전한 64bit 구성방식을 실현한 첫 PowerPC 계렬번호이다.
- **740/740:** G3 처리장치로 알려 졌다. 이 처리장치는 한 소편에 2 준위캐쉬를 집적하고 있으며 무소편캐쉬구성인 처리장치들에 비하여 충분히 성능을 개선하였다.
- **G4:** 이 처리장치는 처리장치소편의 내부속도와 병렬처리기술을 더욱 높이였다.

참고문헌과 Web 싸이트

IBM700 계렬의 서술은 [BELL71a]에서 찾아 볼수 있다. [SIEW82]에는 IBM360, [BELL78a]에는 PDP-8 과 기타 DEC 콤퓨터들의 좋은 적용실례들이 소개되여 있다. 이 세개의 책들에는 또한 1980 년대 초까지 콤퓨터들의 력사를 보여 주는 다른 콤퓨터들에 대한 여러가지 실례들을 상세하게 소개하였다. 전 세대콤퓨터들을 학습하기 위한 가장 좋은 실례들은 [BLAA97]에 소개되였다. 극소형처리장치의 흥미 있는 력사는 [BETK97]에 있다. Pentium 의 가장 좋은 사용법들중의 하나는 [SHAN98]에 있다. 인텔자체의 문서인 [INTE98a,INTE98b]도 좋은 참고문헌이다. [BTEY97]은 32bit 극소형콤퓨터들을 기본으로 하면서 인텔극소형처리장치계렬을 개괄하였다. [IBM94]는 PowerPC 구성방식에 대하여 충분히 고찰하였다. [SHAN95]는 우와 류사한 적용실례와 함께 601 에 대하여 서술하였다. [WEIS94]는 POWER 와 Power PC 구성방식을 취급하였다. 무어의 법칙에 대한 흥미 있는 론의와 그 결과들은 [HUTC96], [SCHA97], [BOHR98]에서 찾아 보면 된다.

BELL71a Bell, C., and Newell, *A Computer Structures: Readings* and Examples. New York: McGraw- Hill, 1971.

BELL78a Bell, C.; Mudge, J.; and McNamara, J. *Computer Engineering*: A DEC *View of Hardware Systems Design*. Bedford, MA: Digital Press, 1978.

BETK97 Betker, M.: Fernando, J.;and Whalen, S."The History of the Microprocessor." *Bell Labs Technical Journal*, Autumn 1997.

BLAA97 Blaauw, G., and Brooks, F. *Computer Architecture: Concepts and Evolution.* Reading, MA:Addison-Wesley, 1997.

BOHR98 Bohr, M. "Silicon Trends and Limits for Advanced Microprocessors." *Communications of The ACM*, March 1998.

BREV97 Brey, B *The Intel Microprocessors: 8086/8066, 80186/80188, 80286, 80386, 80486, Pentium, and Pentiumprocessor*. Upper Saddle river, NJ: Prentice Hall, 1997.

HUTC96 Hutchson, G., and Hutcheson,j." Technology and Economics in The Semiconductor Industry." *Scientific American*, January 1996.

IBM94 International Business Machines, Inc. *The Powerpc Architecture: A Specification for a New Family of Risc Processors*. San Francisco,CA: Morgan Kaufmann, 1994.

INTE98a Intel Corp. *Pentium Processors and Related Products*. Aurora, CO, 1998.

INTE98b Intel Corp. *Pentium Pro and Pentium II Processors and Related Products*. Aurora, CO, 1998.

SCHA97 Schaller, R." Moore's Law: Past, Present, and Future."*IEEE Spectrum*, June 1997.

SHAN98 Shanley, T. *Pentium Pro and Pentium II System Architecture*. Reading, MA: Addison-Wesley, 1998.

SHAN95 Shanley, T. *Powerpc System Architecture*. Reading, MA: Addison-Wesley, 1995.

SIEW82 Siewiorek, D.,;Bell, C.;and Newell, A. *Computer Structures: Principles and Examples*. Newyork: Mcgraw-Hill, 1982.

WEIS94 Weiss, S., and Smith, J. *Power and Powerpc*. San Francisco:morgan Kaufmann, 1994.

Web 싸이트:

- Charl Babbage Institute: 콤퓨터의 력사를 취급하는 여러개의 Web싸이트에 대한 련결을 준다.
- Power PC: Power PC에 대한 IBM의 홈페지
- Developer Home: 개발자들을 위한 인텔의 Web 페지: Pentium 정보를 접근하기 위한 시작점을 준다.

련습문제

1. C(I)=A(I)+B(I)를 구하기 위한 1000개의 마디를 가지는 1차원 A=A(1),A(2)...… A(1000)과 B=B(1), B(2)...…B(1000) 이 있다. IAS명령모임을 리용하여 이 문제를 풀기 위한 프로그람을 작성하시오.
2. IBM360형은 65와 75에서 주소는 두개의 갈라 진 주기억기에서 왔다 갔다한다(즉 우수주소는 한 기억기, 기수주소는 다른 기억기에). 이 기술의 목적은 무엇인가?

제 2 편. 콤퓨터체계

제 2 편의 중심

콤퓨터체계는 처리장치, 기억기, I/O 그리고 기본구성요소들사이의 호상접속으로 이루어 져 있다. 제 2 편에서는 이 구성요소중 제 3 편에서 취급되는 가장 복잡한 처리장치를 제외한 나머지부분들에 대하여 상세히 고찰한다.

제 2 편에서는 두가지 기본문제를 론의한다.

제 2 편의 장별 내용

제 3 장. 체계모선

콤퓨터체계에서 매개 기본구성요소들의 총체적인 기능, 호상접속구조, 그것들사이에 교환되는 신호형태들을 정의하는것은 콤퓨터체계에 대한 가장 높은 준위의 서술이다. 제 3 장에서는 호상접속구조와 그 구조를 리용한 신호교환에 대하여 고찰한다. 콤퓨터체계의 기본구성요소들을 호상접속하는 가장 일반적인 수단들은 체계모선이나 모선묶음이며 이것이 바로 제 3 장의 중심이다. 또한 이 장에서는 호상접속설계에 영향을 주는 기본지표들, 특수하게 새치기를 지원하는 요구들을 고찰한다.

제 4 장. 내부기억기

이 장에서는 주기억기의 구성과 성능을 높이기 위한 캐쉬전략의 리용에 대하여 고찰한다. 주기억기체계의 설계는 큰 기억용량, 짧은 접근시간, 저가격과 같은 세가지 경제적인 설계요구들사이의 끊임 없는 충돌과정이다. 기억기기술의 발전에 따라 이 3가지 특성들은 계속 변화하므로 주기억기구성에서 설계지도서들을 새롭게 작성하고 검사하여야 한다. 제 4 장에서 특별히 강조된 두개의 련관되는 부분은 캐쉬구성과 DRAM을 위한 여러가지 방안들이다.

제 5 장. 외부기억기

주기억기에서 리용할수 있는것보다 실제로 큰 기억용량을 가지며 더 영구적으로 기억할수 있는 외부기억기를 구성할것을 요구한다. 가장 널리 리용되는 외부기억기의 형태는 자기디스크이며 제 5 장에서는 대부분 이것을 기본으로 취급한다. 첫째로, 자기디스크기술과 설계를 고찰한다. 둘째로, 디스크기억기성능을 개선하기 위한 RAID 구성의 리용을 고찰한다. 또한 제 5 장에서는 빛기억기와 자기테프도 고찰한다.

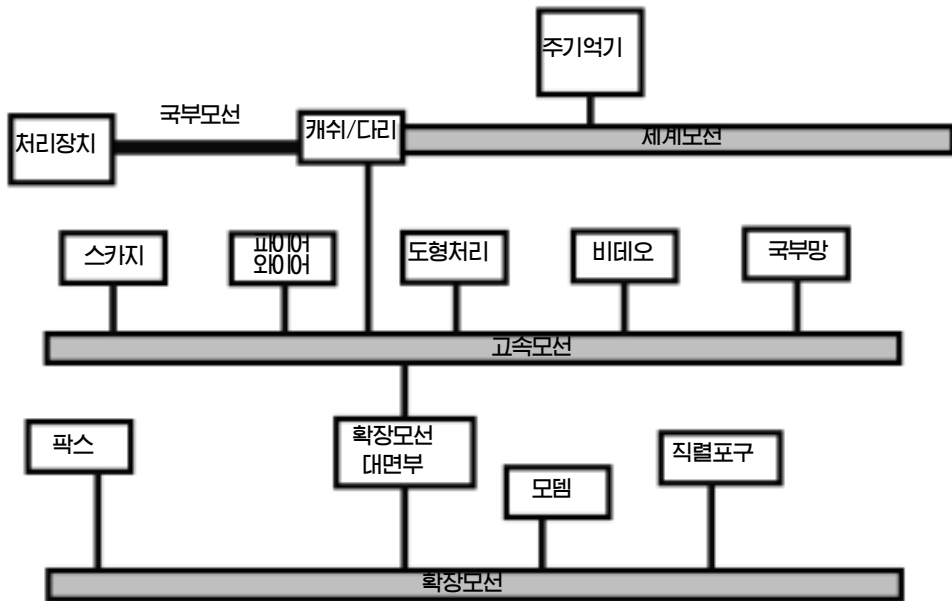
제 6 장. 입출력

제 6 장에서는 I/O 구성의 여러가지 형태를 기본적으로 고찰하였다. 이 부분은 복잡하므로 요구하는 성능을 만족시키기 위한 콤퓨터체계설계의 다른 부분들보다 리해하기 어렵다. 제 6 장에서는 프로그람식 I/O, 새치기 I/O, 직접접근(DMA)의 기술들을 리용함으로써 I/O 장치가 콤퓨터체계의 휴식상태들을 서로 리용할수 있게 하는 기구들을 고찰한다. 또한 I/O 장치와 외부장치사이대면부를 고찰한다.

제 7 장. 조작체계

조작체계에 대한 상세한 설명은 이 책에서 하지 않는다. 그러나 조작체계의 기본기술을 리해하며 요구하는 성능을 얻기 위하여 OS 가 하드웨어를 어떻게 잘 리용해야 하는가 하는것을 알아야 한다. 제 7 장은 조작체계의 기본원리와 조작체계지원을 위한 콤퓨터하드웨어의 특수한 설계특성을 서술한다.

제 3 장. 체계모선

◆◆

- 명령주기는 명령꺼내기, 0 이상의 연산수꺼내기, 0 이상의 연산수기억, 새치기검사(만일 새치기가 가능하다면)로 구성된다.
- 기본콤퓨터체계구성요소(처리장치, 주기억기, I/O 장치)들은 자료와 조종신호들을 교환하는 순서대로 서로 접속될것을 요구한다. 호상접속의 가장 일반적인 방법은 여러개의 선들로 구성된 공유된 체계모선을 리용하는것이다. 현대체계들에서는 일반적으로 성능개선을 위하여 모선들의 계층화를 실현한다.
- 모선을 위한 기본설계요소에는 중재(모선에 신호들을 보내는 허가를 중심적 혹은 분산적방법으로 조종하는), 동기(모선에서의 신호들이 중심박자에 동기되든지 혹은 가장 최근의 전송에 기초하여 비동기적으로 전송하든지), 모선폭(주소모선의 수와 자료모선의 수)이 속한다.

◆◆

높은 준위에서 고찰할 때 콤퓨터는 CPU, 기억기 그리고 I/O 구성요소들을 하나이상씩 가지고 있다. 이 구성요소들은 프로그람을 집행하기 위한 콤퓨터의 기본기능들을 수행하도록 여러가지 방법으로 호상 접속된다. 따라서 콤퓨터체계를 높은 준위에서 고찰하려면 (1)구성요소들의 외적인 거동 즉 다른 구성요소들과 자료와 조종신호들의 호상교환 (2)이 구성요소들의 호상간 접속구조로 나누어 서술할수 있다.

높은 준위에서 구조와 기능을 고찰하는것은 콤퓨터의 본질을 리해하는데서 설득력이 있는것으로 하여 중요하다. 성능평가의 복잡한 항목들을 하나씩 하나씩 리해해 나가는 것도 역시 중요하다. 높은 준위의 구조와 기능을 파악하면 체계성능제한요소, 어떤 요소가 고장났을 때에 바꾸어 리용할수 있는 여러가지 통로, 체계고장의 범위를 인차 알수 있게 하며 성능을 추가적으로 강화할수 있다. 많은 경우에 체계의 위력과 고장안정성을 높일데 대한 요구는 단순히 개별적구성요소들의 속도와 믿음성을 증가시키는 방법으로가 아니라 설계를 변화시키는 방법으로 만족시킨다.

이 장은 콤퓨터구성요소들의 호상접속에 리용되는 기본구조를 중심으로 고찰한다. 이 장에서는 기본구성과 그것들사이의 호상접속요구들을 간단히 고찰하는것으로부터 시작한다. 그다음에 기능들을 고찰한다.

다음으로 체계구성요소들을 호상 접속하기 위한 모선들의 리용을 고찰한다.

제 1 절. 콤퓨터구성요소

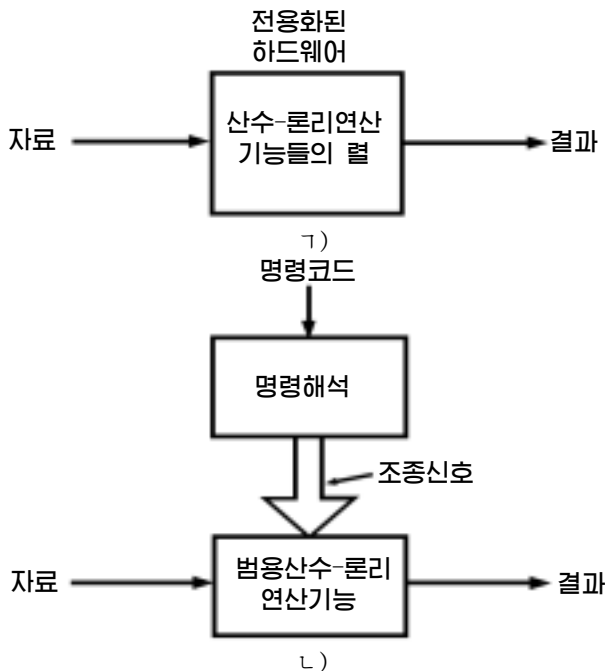
제 2 장에서 론의한것처럼 실제적으로 모든 현대콤퓨터설계들은 존 폰 노이만에 의하여 개발된 개념들에 기초하고 있다. 이 설계는 폰 노이만방식으로 알려 져 있는 몇가지 기본개념에 기초한다.

- 자료와 명령들은 하나의 읽기/쓰기기억기에 기억된다.
- 이 기억기의 내용들은 거기에 포함된 자료형식에 관계없이 위치에 따라 주소화된다.
- 실행은 임의의 명령으로부터 다음명령에로 순차적인 방법으로 진행한다.

그림 3-1. 하드웨어와 쏘프트웨어의 접근방법
ㄱ-하드웨어에서 프로그람화, ㄴ-쏘프트웨어에서 프로그람화

이 개념들을 제 1 장에서도 고찰하였으나 여기서 간단히 고찰한다. 이것은 2 진수자료를 기억하며 그 자료에 대하여 산수연산과 론리연산을 수행할수 있게 여러가지 방법들로 련결할수 있는 기본론리적구성요소들의 작은 모임이다. 만일 특수한 계산을 집행하여야 한다면 이 계

산을 위하여 특수하게 설계된 론리적구성요소들을 배렬할수 있다. 프로그람작성의 형식에 따라 요구되는 배렬에서 여러가지 구성요소들을 호상 련결하는 과정을 생각할수 있다. 결과 《프로그람》은 하드웨어형이며 **하드웨어프로그람**이라고도 한다.

이제 이것들중 어느 하나를 고찰하자. 산수론리기능의 일반배렬을 구성한다고 하자. 하드웨어의 이 묶음은 하드웨어에 적용된 조종신호들에 의존하는 자료들에 대한 여러가지 기능을 수행한다. 전용화된 하드웨어였던 최초의 경우 체계는 자료를 받아 들여 결과를 만든다(그림 3-1 ㄱ). 일반용하드웨어에서 체계는 자료와 조종신호를 받아 들이고 결과를 생성한다. 따라서 매개의 새로운 프로그람을 위한 하드웨어의 재배선대신에 프로그람작성자는 단지 새로운 조종신호선묶음을 공급할것을 요구한다.

얼마나 적은 조종신호들이 공급되는가? 대답은 간단하다. 그러나 명백치 않다. 전반적인 프로그람은 실제적으로 집행단계의 순서이다. 매 단계에서 조종신호의 새로운 묶음들이 요구된다. 조종신호의 가능한 모든 모임들을 위한 유일코드를 제공하고 이 코드를 입력하여 조종신호를 발생하는 부분을 일반용하드웨어에 추가한다(그림 3-1 ㄴ).

프로그람작성은 오늘에 와서 더 쉬워 졌다. 매번 새로운 프로그람을 위하여 하드웨어를 교체하지 않고 새로운 코드렬을 제공하기만 하면 된다. 결과 매개 코드는 명령이며 하드웨어의 부분은 매 명령을 해석하여 조종신호를 발생한다. 프로그람작성의 이 새로운 방법을 구별하기 위하여 코드들의 렬 즉 명령을 **쏘프트웨어**라고 한다.

그림 3-1 ㄴ는 체계의 두 중요구성요소 즉 명령해석기와 일반목적산수-론리기능장치를 보여 준다. 이 두개가 CPU를 구성한다. 그리고 콤퓨터가 기능을 수행하도록 하기 위하여 이밖의 몇가지 구성요소들이 더 요구된다. 자료와 명령들이 체계에 넣어 져야 한다. 이를 위하여 몇가지 종류의 입력장치가 필요하다. 이 장치는 어떤 형태로 자료와 명령을 입력하고 체계에서 리용할수 있는 신호의 내부형식으로 변환하기 위한 기본구성

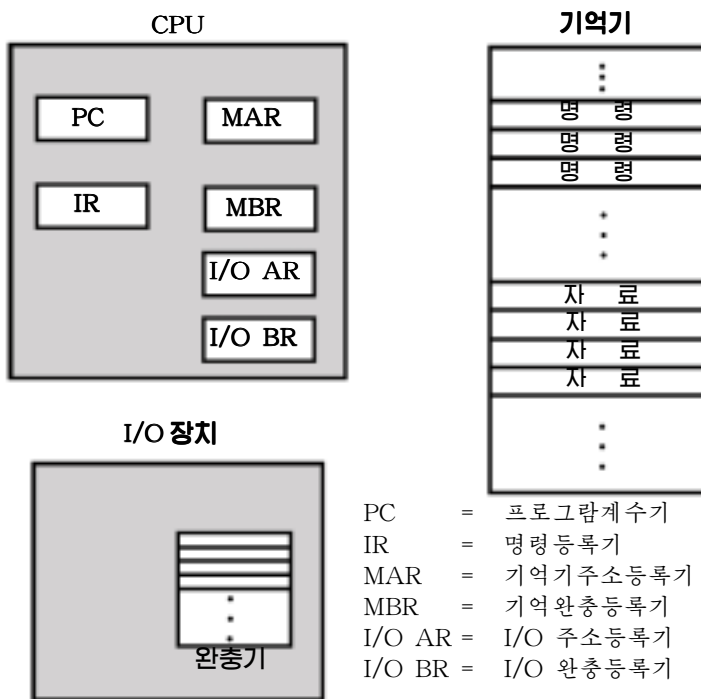



그림 3-2. 콤퓨터구성요소:웃준위고찰

요소들로 이루어 진다. 또한 결과를 보고하는 수단이 요구되는데 이것은 출력장치에 의하여 실현된다. 바로 이 두개를 합쳐 **I/O 구성요소**이라고 한다.

또 하나의 구성요소이 요구된다. 입력장치는 련달아 명령과 자료를 나른다. 그러나 프로그람은 변함없이 차례로 집행되지 않는다. 즉 이행이 가능하다(실례로 IAS 이행명령). 마찬가지로 자료에 대한 연산도 미리 정한 순서로 한번에 한개요소이상을 접근할 것을 요구한다. 따라서 명령과 자료들을 림시적으로 기억시키기 위한 공간이 있어야 한다. 이 장치를 외부기억기 혹은 주변장치들과 구별하기 위하여 기억기 혹은 주기억기라고 한다. 폰 노이만은 하나의 기억기가 명령과 자료를 둘다 기억하게 하였다.

그림 3-2는 높은 준위구성요소들과 그것들사이의 호상작용을 보여 준다. CPU는 기억기와 자료를 교환한다. 이를 위하여 일반적으로 CPU내에 있는 두개의 내부등록기 즉 다음의 읽기나 쓰기를 위하여 기억기주소를 지정하는 기억기주소등록기(MAR)와 기억기에 써질 자료를 가지거나 기억기로부터 읽은 자료를 받는 기억기완충등록기(MBR)를 리용한다. 마찬가지로 I/O 주소등록기(I/OAR)는 개별적인 I/O 장치를 지적한다. I/O 완충등록기는 I/O 장치와 CPU 사이에 자료를 교환하기 위하여 리용한다.

기억기모듈은 위치들의 묶음으로 이루어 지며 차례로 번호가 붙은 주소들로 정의된다. 매개 위치는 명령이나 자료로 해석될수 있는 2진수를 가지고 있다. I/O 장치는 외부장치로부터 CPU와 기억기에로 혹은 CPU와 기억기에서 외부장치에로 자료를 전송한다. 이 장치는 목적지에 자료를 전송할 때까지 이것을 림시적으로 유지하기 위한 내부완충기를 가지고 있다.

여기서는 기본구성요소들에 대한 간단한 고찰과 함께 이 구성요소들이 프로그람들을 어떻게 집행시키는가를 개괄한다.

제 2 절. 콤퓨터의 기능

콤퓨터에 의하여 수행되는 기본기능은 기억기에 기억된 명령들의 모임으로 구성된 프로그람에 의하여 집행된다. 처리장치는 프로그람에 지적된 명령들을 실행함으로써 실제적으로 일을 한다. 이 절에서는 프로그람집행의 기본요소들을 고찰한다. 가장 간단한 형태는 명령처리가 두 단계로 이루어 진 경우이다. 즉 처리장치는 한번에 기억기로부터 명령을 꺼내며 매 명령을 실행한다. 프로그람집행은 명령꺼내기와 명령실행의 반복으로 이루어 진다. 명령실행은 여러가지 연산을 포함하며 명령의 내용에 관계된다(실례로 그림 2-4의 아래부분 참고).

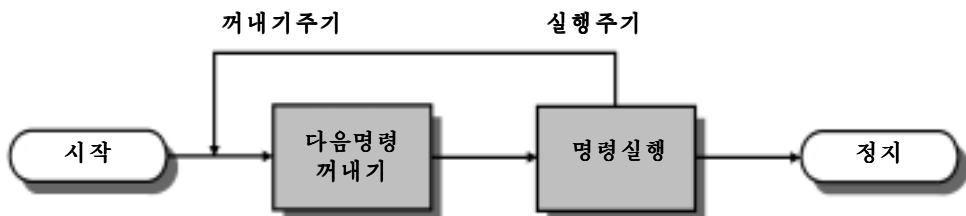



그림 3-3. 기본명령주기

한개의 명령을 처리하기 위하여 요구되는 시간을 **명령주기**라고 한다. 앞에서 고찰한 간단한 두 단계의 분류를 리용하면 명령주기화는 그림 3-3 에서와 같이 주어 진다. 두 단계를 **꺼내기주기**와 **집행주기**라고 한다. 프로그람집행은 처리장치가 동작하지 않을 때와 일부 치명적인 고장이 발생하였을 때 혹은 콤퓨터를 정지시키는 명령이 실행되고 있는 동안 정지된다.

1. 명령꺼내기와 실행

매 명령주기가 시작되면 처리장치는 기억기로부터 명령을 꺼낸다. 일반적인 처리장치의 프로그람계수기(PC)는 다음에 꺼낼 명령의 주소를 가지고 있다. 다시말하여 처리장치는 다음명령을 순서대로 꺼내기 위하여 매 명령을 꺼낸후에 PC 를 항상 증가시킨다. 그러한 실례로서 매 명령이 기억기의 16bit 단어를 차지하는 콤퓨터를 고찰하자. 프로그람계수기가 위치 300 으로 설정되였다고 가정하자. 계속되는 명령주기에서 처리장치는 301, 302, 303 등의 위치로부터 명령들을 꺼낼수 있다. 이 순서는 앞으로 설명되는것처럼 변화된다.

꺼내여 진 명령은 명령등록기(IR)로 알려 진 처리장치의 등록기에 넣어 진다. 명령은 처리장치가 가지고 있는 동작을 지적하는 비트들을 가지고 있다. 처리장치는 명령을 해석하고 요구되는 동작을 실행한다. 일반적으로 이 동작들은 4 개의 종류로 규정된다.

- **처리장치-기억기:** 자료는 처리장치로부터 기억기로, 기억기로부터 처리장치까지 전송된다.
- **처리장치-I/O:** 자료는 처리장치와 I/O 장치사이의 자료전송으로서 주변장치에로 또는 주변장치로부터 전송된다.
- **자료처리:** 처리장치는 자료에 대하여 일부 산수 혹은 론리연산을 수행한다.
- **조종:** 어떤 명령은 실행순서가 변하도록 지적한다. 실례로 처리장치가 다음명령이 위치 182에 있다는것을 지적하는 149위치의 명령을 꺼냈다고 하자. 처리장치는 182로 PC 를 설정함으로써 이것을 실행한다. 따라서 다음꺼내기주기에서 명령은 150 이 아니라 위치 182 로부터 꺼내여 진다.

하나의 명령의 실행은 이 동작들의 접속을 다 포함한다.

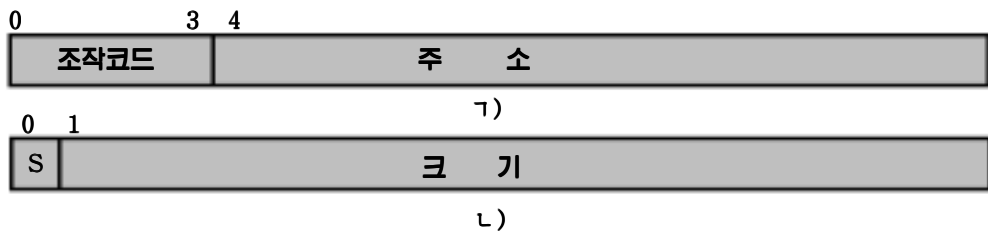



그림 3-4. 가정적인 기계의 특징

ㄱ-명령형식, ㄴ-옹근수형식, ㄷ-내부 CPU 등록기, ㄹ-조작코드의 실례

그림 3-4 에 보여 준 특징들을 포함하는 가정적인 처리장치를 리용한 간단한 실례를 고찰하자. 처리장치는 축적기라고 하는 하나의 자료등록기를 포함한다. 명령과 자료는 16bit 길이이다. 따라서 16bit 단어들을 리용하여 기억기를 구성하는것이 편리하다. 명령형식은 조작코드부로 4bit 를 제공하며 따라서 $2^4 = 16$ 개의 서로 다른 조작코드를 가질수 있다. 그리고 직접주소화할수 있는 $2^{12} = 4096$(4K)기억기를 지적할수 있다.

그림 3-5 에서는 관련 있는 기억기일부와 처리장치등록기들, 국부적인 프로그람집행을 설명하였다[1]. 이 프로그람은 주소 940 의 기억기단어의 내용을 주소 941 의 기억기단어의 내용에 더하여 결과를 주소 941 주소에 기억하는것을 보여 준다. 세개의 명령꺼내기주기와 세개의 실행주기로 구성되는 세개의 명령이 요구된다.

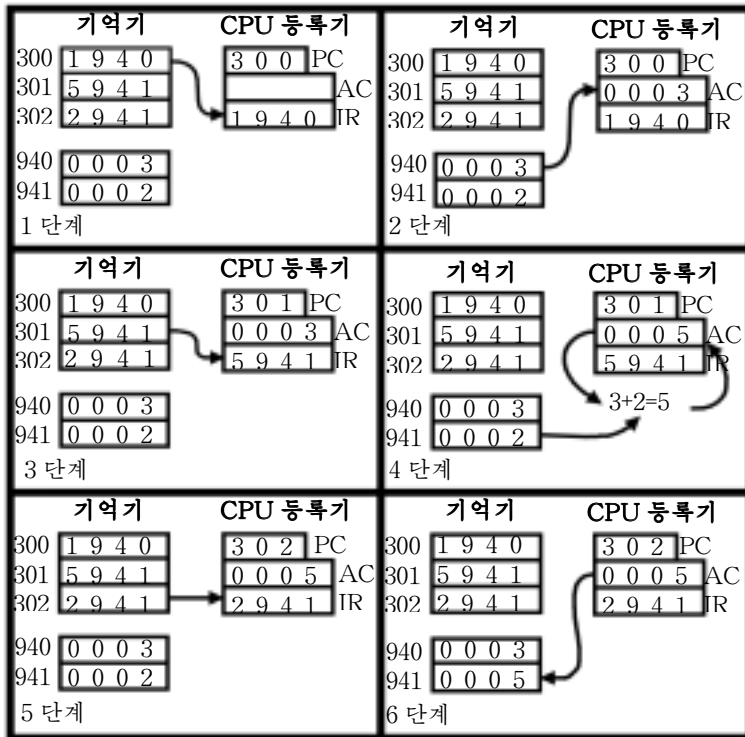



그림 3-5. 프로그람실행의 실례

- PC 는 300 이며 첫번째 명령의 주소이다. 이 명령은 명령등록기 IR 에 전송된다.
- IR 에서 첫 4bit 는 축적기에 대한 전송이라는것을 지적하며 나머지 12bit 는 주소 940 이라는것을 지적한다.
- PC 는 증가하며 다음명령을 꺼낸다.
- AC 의 내용과 주소 941 의 내용은 더해 지며 결과는 AC 에 기억된다.
- PC 는 증가하며 다음명령을 꺼낸다.
- AC 의 내용이 주소 941 에 기억된다.

이 실례에서 매개 명령꺼내기주기와 명령실행주기로 이루어 진 세개의 명령주기들은 주소 941 의 내용에 주소 940 의 내용을 더할것을 요구한다. 명령들이 더 고급화되고

---

[1] 매개 수자를 4bit 로 표시하는 16 진표기가 리용된다. 이것은 단어길이가 4 의 배수인 경우 기억기와 등록기의 내용을 표시하는 가장 일반적인 표시법이다. 16 진표기법은 제 8 장 부록에서 고찰하게 된다.

더 빨리 실행될것을 요구한다. 현대처리장치들은 하나이상의 주소를 가지는 명령들을 가지고 있다. 따라서 개별적인 명령에 대한 실행주기는 한번이상 기억기를 참조한다. 또한 기억기참조대신에 어떤 명령은 I/O 조작을 지정한다.

실례로 기호언어 ADD A,B 로 표시된 PDP-11 명령은 기억기 B 와 A 의 내용의 합을 기억주소 A 에 기억한다. 한개 명령주기는 다음과 같은 단계로 주어 진다.

- ADD 명령꺼내기
- A 주소의 내용을 처리장치에로 읽어 들이기
- B 주소의 내용을 처리장치에로 읽어 들이기. 처리장치는 A 의 내용을 잃어 버리지 않고 기억기값을 보존하기 위하여 한개의 축적기외에 적어도 두개의 등록기를 가져야 한다.
- 두 값을 더한다.
- 주소 A 에 처리장치로부터 결과를 쓴다.

따라서 개별적인 명령들에 대한 실행주기는 한번이상 기억기를 참조한다. 또한 기억기참조대신 어떤 명령은 I/O조작을 지정한다. 이 추가적인 기능을 더 충분히 고찰하자. 그림 3-6 은 그림 3-3 의 기본명령주기를 더 상세히 보여 준다. 이 그림은 상태도의 형태로 주어 졌다. 임의의 주어 진 명령주기에 대하여 몇개의 상태는 무효상태이고 다른것들은 하나이상의 다른 상태로 넘어 간다. 상태들은 다음과 같이 서술된다.

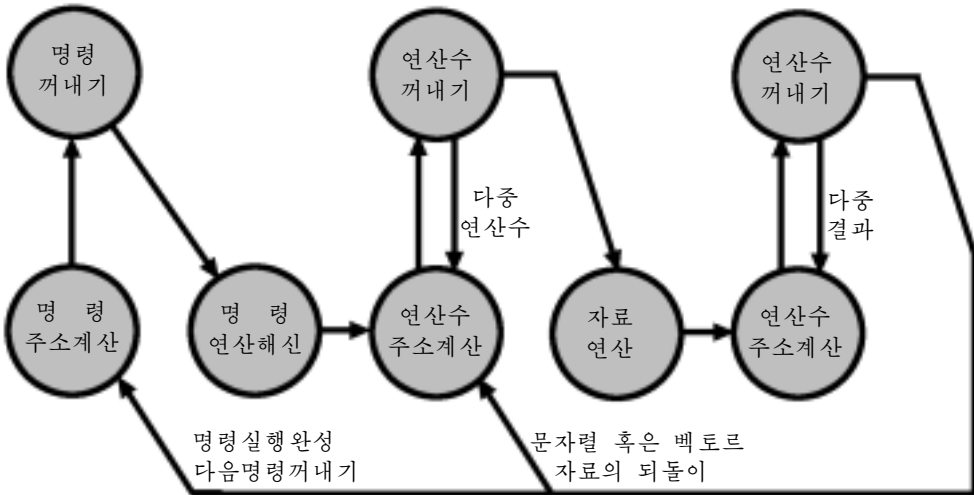



그림 3-6. 명령주기상태도

- **명령주소계산(iac):** 실행을 위한 다음의 주소를 결정한다. 보통 앞의 명령의 주소에 고정된 수들을 더한다. 실례로 만일 매 명령이 16bit 길이이고 기억기가 16 bit 단어로 구성되였다면 이때 앞의 주소에 1 을 더한다. 또한 기억기가 개별적인 주소에 대하여 8bit 로 구성되였다면 앞의 주소에 2 를 더한다.
- **명령꺼내기(if):** 기억기로부터 명령을 처리장치로 읽어 들인다.
- **명령조작코드해신(iod):** 수행하는 연산과 연산에 리용되는 연산수들의 형식을 알기 위하여 명령을 분석한다.

- **연산수주소계산(oac):** 연산이 기억기 혹은 I/O 를 통하여 연산수의 참조를 가진다면 연산수의 주소를 결정한다.
- **연산수꺼내기(of):** 기억기로부터 연산수를 꺼내거나 I/O 로부터 연산수를 읽는다.
- **자료연산(do):** 명령에서 지적된 연산을 지정한다.
- **연산수기억(os):** 기억기에 결과를 쓰거나 I/O 로 출력한다.

그림 3-6의 앞부분에서 상태는 처리장치와 기억기 혹은 I/O장치사이의 교환을 반영한다. 그림의 뒤부분에서 상태는 처리장치연산을 반영한다. oac 상태는 명령이 읽기, 쓰기 혹은 둘 다 포함할수 있으므로 두가지를 나타낸다. 그러나 이때 수행되는 동작이 두 경우에 대하여 상태가 같으며 따라서 하나의 상태로 통일시킬것을 요구한다.

또한 그림은 같은 처리장치에서 일부 명령이 다중연산수, 다중결과를 요구하므로 이것을 실행할수 있게 반영하였다. 실례로 PDP-11 명령 ADD A, B 는 상태들의 다음의 순서 즉 iac, if, iod, oac, of, oac, of, do, oac, os 를 나타낸다.

결국 같은 처리장치에서 한개의 명령은 문자들의 렬 혹은 수들의 벡토르로 수행되는 연산을 지적할수 있다. 그림 3-6 에 보여 준것처럼 이것은 연산수의 꺼내기와 보관에 대한 연산들을 반복할수 있다.

2. 새치기

실제적으로 모든 콤퓨터들에는 다른 장치(I/O, 기억기)가 처리장치의 일반적인 순서를 새치기하도록 하는 기구들이 있다. 표 3-1 은 가장 일반적인 새치기종류를 보여 준다. 이 새치기들에 대한 명백한 고찰은 이 책의 뒤부분에서 진행하며 특별히 제 6 장과 제 11 장에서 론의한다. 그러나 지금은 명령실행의 내용과 호상접속구조에서 새치기들의 밀접한 관계를 더 명백히 리해하기 위한 개념만을 고찰한다. 독자들은 새치기들의 발생과 처리의 세부에 대한 단계들을 알고 있으므로 새치기로부터 초래되는 장치들사이의 통신을 기본으로 하여 고찰한다.

새치기는 기본적으로 능률적인 처리를 제공하기 위한 수단으로 리용된다. 실례로 대부분 외부장치들은 처리장치보다 속도가 뜨다. 처리장치가 그림 3-3 의 명령주기단계들을 리용하여 인쇄기에 자료를 전송하고 있다고 가정하자. 매 쓰기연산후에 처리장치는 정지되여야 하며 인쇄기가 자료를 받을 준비가 될 때까지 휴식상태를 유지하여야 한다. 이 정지시간의 길이는 기억기를 포함하지 않는 명령주기를 수백 혹은 수천개 실행할수

표 3-1. 새치기의 종류

프로그람새치기	산수연산자리넘침, 령나누기오유, 비법적인 기계어명령실행의 영향, 사용자용기억기공간외부참조와 같은 명령실행의 결과로 나타나는 일부 조건들에 의하여 발생
시간새치기	처리장치안의 시계에 의하여 발생. 이 새치기는 조작체계가 규칙적인 확실한 기능들을 수행하도록 허가
I/O 새치기	일반적으로 조작의 완성을 신호하거나 여러가지 오유조건들을 신호하기 위하여 I/O 로부터 발생
장치오유새치기	전원고장이나 기억기기우성오유와 같은 오유에 의하여 발생

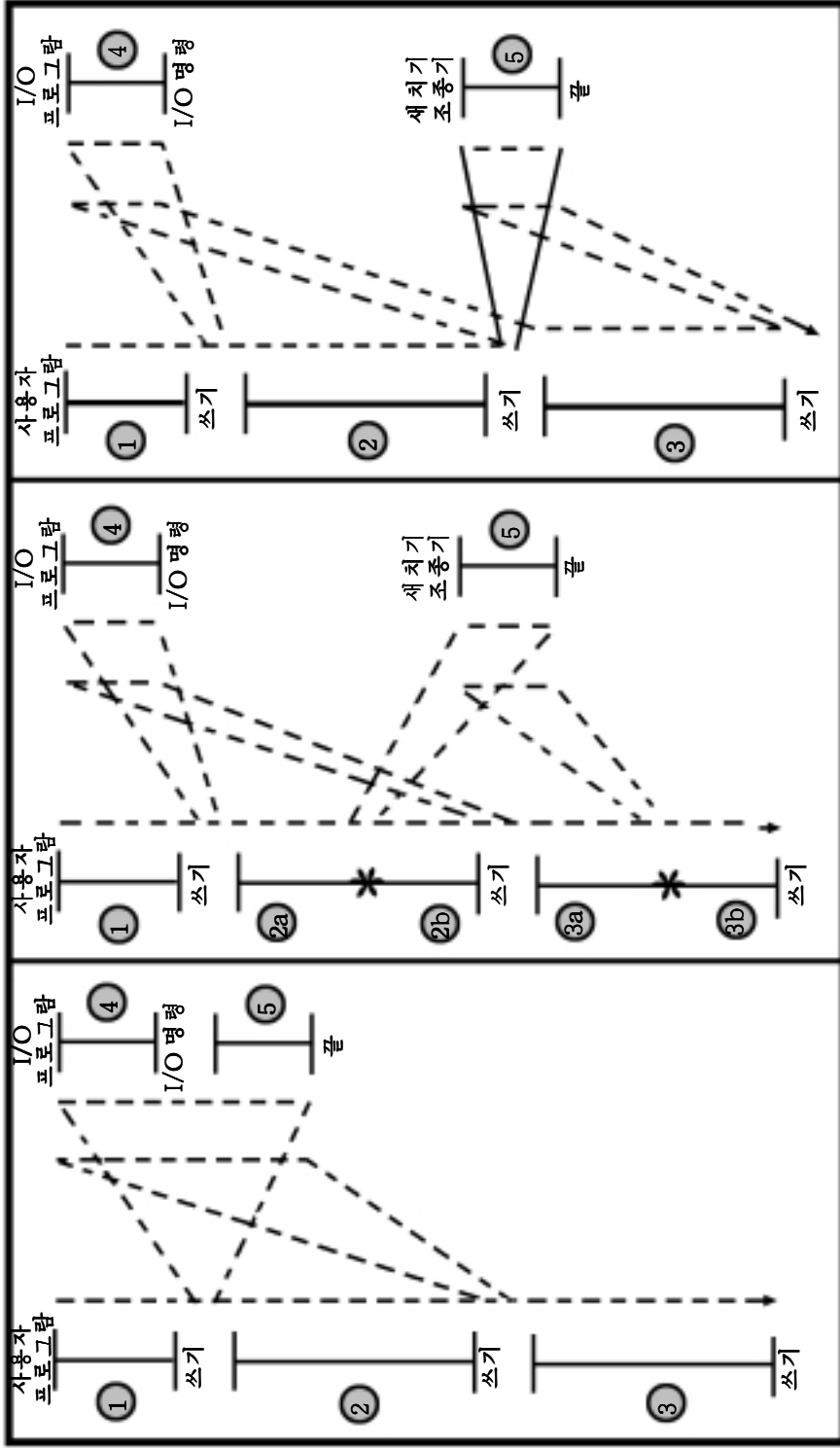



그림 3-7. 새치기 없는 경우와 있는 경우의 프로그람조종흐름
ㄱ-새치기를 리용하지 않은 경우, ㄴ-새치기경우, 짧은 I/O 기다림, ㄷ-새치기경우: I/O 기다림

있다. 명백히 이것은 처리장치의 사명을 무의미하게 한다.

그림 3-7 ㄱ에 요인사건들의 상태를 설명하였다. 사용자프로그람은 처리도중에 삽입하는 WRITE 접근렬을 수행한다. 코드토막 1, 2 와 3 은 I/O 를 포함하지 않는 명령들의 순서렬이라고 할수 있다. WRITE 접근들은 체계봉사프로그람이며 실제로 I/O 조작을 진행하는 I/O 프로그람을 접근한다. I/O 프로그람은 세개의 부분으로 구성된다.

- 그림에서 4로 표시된 실제적인 I/O조작을 준비하기 위한 명령들의 렬. 이것은 지정된 완충기에로 출력하기 위한 자료복사와 장치에 지령을 주기 위한 파라메터들의 준비를 포함한다.
- 실제적인 I/O 명령. 새치기의 리용이 없이 이 지령이 일단 집행되면 프로그람은 요구되는 기능을 수행하기 위한 I/O 장치를 위하여 기다려야 한다. 프로그람은 I/O 조작이 완료되였는가를 결정하기 위한 검사연산을 반복적으로 수행하면서 기다려야 한다.
- 그림에서 5 로 표시된 명령들의 렬. 이것은 연산을 완성하기 위한것이다. 즉 연산이 성공인가 실패인가를 지적하는 기발설정을 진행한다.

I/O 조작은 완료될 때까지 상대적으로 오랜 시간이 걸리므로 I/O 프로그람은 연산이 완성될 때까지 오래 기다려야 한다. 결국 사용자프로그람은 많은 시간 WRITE 접근의 점에서 정지된다.

새치기와 명령주기

새치기로 I/O 조작을 진행하는 동안 처리장치가 다른 명령들을 실행하도록 하게 할수 있다. 그림 3-7 ㄴ에서 조종흐름을 고찰하자. 앞에서처럼 사용자프로그람은 WRITE 접근형식으로 체계접근을 진행하는 위치에 도달한다. 이 경우에 I/O 프로그람은 준비코드와 실제I/O명령을 불러 낸다. 이 적은 명령들이 실행된후에 조종은 사용자프로그람에로 돌아 온다. 한편 외부장치는 콤퓨터기억기로부터 자료를 접수하여 그것을 인쇄하는 동작을 진행한다. I/O 조작은 사용자프로그람에서 명령들의 실행과 동시에 진행된다.

외부장치가 봉사될 준비가 되면 즉 처리장치로부터 자료들을 접수할 준비가 되면 그 외부장치를 위한 I/O 장치는 처리장치에 **새치기요구**신호를 보낸다. 처리장치는 현재 프로그람의 연산을 중단하고 I/O 장치를 봉사하기 위한 프로그람에로 이행하며 그 장치가 봉사된후에 원래의 실행을 다시 하게 된다. 이와 같은 새치기들이 발생한 시점은 그림 3-7 ㄴ에서 별표로 표시되였다.

사용자프로그람의 준위에서 고찰해 보면 새치기는 집행의 일반순서에 대한 중단이다. 새치기처리가 완성되였을 때 집행은 원래프로그람에로 되돌아 간다(그림 3-8). 따라서 사용자프로그람은 새치기들을 조종하기 위한 어떤 지적된 코드를 포함하지 않아도 된다. 즉 처리장치와 조작체계는 사용자프로그람을 중단시키고 그다음 같은 위치에로 되돌려 보낸다

새치기를 조종하기 위하여 **새치기주기**는 그림 3-9에서 보여 주는것처럼 명령주기에 포함된다. 새치기주기에서 처리장치는 어떤 새치기가 발생하였는가를 검사한다. 만일 새치기요구가 없다면 처리장치는 꺼내기주기로 넘어 가며 현재프로그람의 다음명령을 깨낸다. 만일 새치기요구가 접수되면 처리장치는 다음과 같이 동작한다.

- 집행되고 있는 현재프로그람의 집행을 중지하고 그 내용을 보관한다. 이것은 집행을 계속하기 위한 다음명령의 주소를 보관한다는것을 의미한다. 또한 처리장치의 현재동작과 관련 있는 다른 자료들도 보관한다.

● 프로그람계수기는 **새치기처리프로그람**의 시작주소로 설정된다.

처리장치는 명령꺼내기주기에로 넘어 가며 새치기를 봉사하는 새치기프로그람에서 첫번째 명령을 꺼낸다. 새치기처리프로그람은 일반적으로 조작체계의 부분프로그람이다. 일반적으로 이 프로그람은 새치기의 내용을 결정하며 요구되는 동작을 수행한다. 실례로 새치기처리프로그람은 어떤 I/O장치가 새치기를 발생했다는것을 알면 그 I/O장치에 더 많은 자료를 쓰기할수 있는 프로그람에로 이행한다. 새치기처리프로그람이 집행을 끝냈을 때 처리장치는 그 새치기처리프로그람으로부터 사용자프로그람집행으로 되돌아 가야 한다.

이 처리에 일정한 시간이 걸린다는것은 명백하다. 특권명령들은 새치기의 내용을 알기 위하여 또한 적당한 동작을 결정하기 위하여 실행된다. 이로부터 I/O 조작에서의 단순한 기다림에 의하여 잃어 버리는 시간이 상대적으로 많으므로 처리장치는 새치기를 리용하여 더 높은 능률을 낼수 있다.

그림 3-8. 새치기에 의한 조종이행

그림 3-9. 새치기를 가진 실행주기

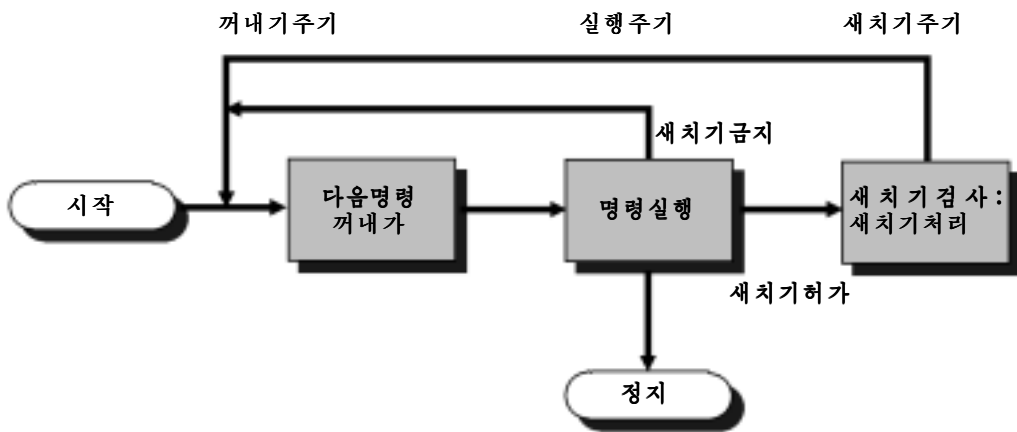
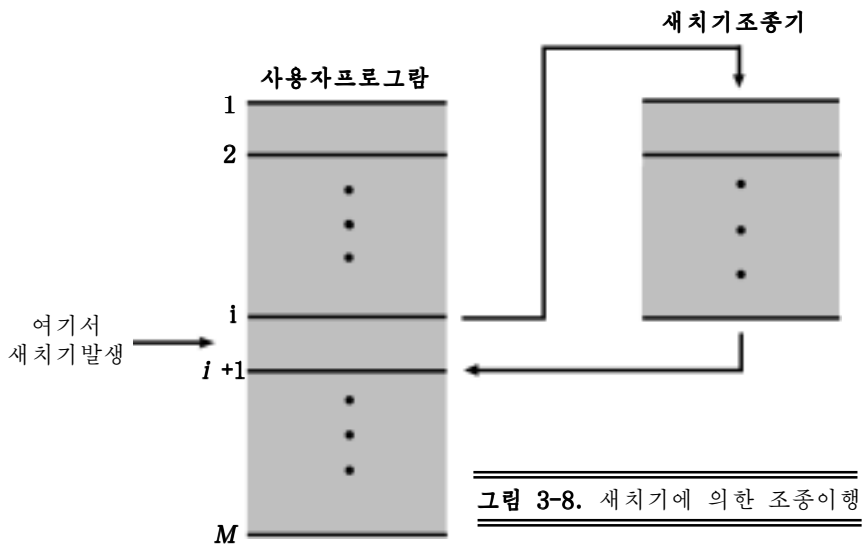
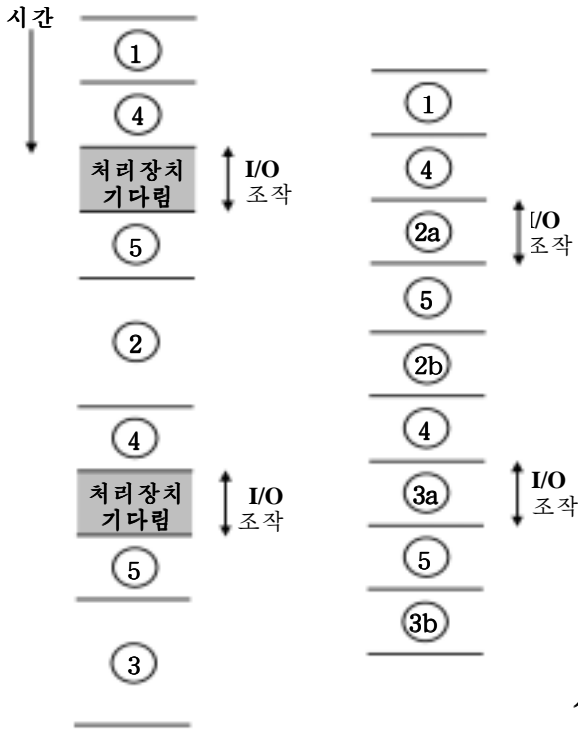



그림 3-10. 프로그람동기: 짧은 I/O 기다림
ㄱ- 새치기 없는 경우, ㄴ- 새치기 있는 경우

그림 3-11 은 새치기리용이 없을 때의 이 조건들에 대한 동기를 보여 준다. I/O 조작에 사용자명령들의 실행과 겹쳐서 병렬로 실행되면 이 겹치는 부분에 의하여 능률이 높아진다는것을 알수 있다.

그림 3-12 에는 새치기주기처리과정을 포함하는 수정된 명령주기상태도를 보여주었다.

다중새치기

지금까지는 단일새치기의 발생에 대해서만 고찰하였다. 그러나 여러개의 새치기가 발생한다고 가정하자. 실례로 프로그람은 통신선으로부터 자료를 받으면서 결과들을 인쇄하여야 한다고 하자. 인쇄기는 인쇄조작을 완성하는 때마다 새치기를 발생할것이다. 통신선조종기는 자료의 한개 단위가 도착할 때마다 새치기를 발생할것이다. 자료단위는 통신선로의 통

능률에서 리득을 고찰하기 위하여 그림 3-7 ㄱ와 그림 3-7 ㄴ에서 조종흐름에 기초한 시간동기를 보여 준 그림 3-10 을 고찰하자. 그림 3-7 ㄴ와 그림 3-10 은 I/O 조작을 위하여 요구되는 시간이 상대적으로 짧다고 가정한다. 즉 사용자프로그람에서 쓰기연산들사이의 명령실행을 완성하는 시간이 매우 짧다. 더 일반적인 경우 특히 인쇄기와 같은 느린 장치에 대하여 I/O조작은 사용자명령들의 순차실행보다 더 많은 시간이 걸린다. 그림 3-7 ㄷ는 사건들의 이 상태를 보여 준다. 이 경우 사용자프로그람은 첫번째 접근에 의한 많은 I/O 조작이 끝나기전에 두번째 WRITE 접근에 이르게 된다. 결과 사용자프로그람이 그 점에서 정지되게 된다. 진행중의 I/O 조작이 완성되면 이 새로운 WRITE 접근은 처리되며 새로운 I/O 조작이 시작된다.

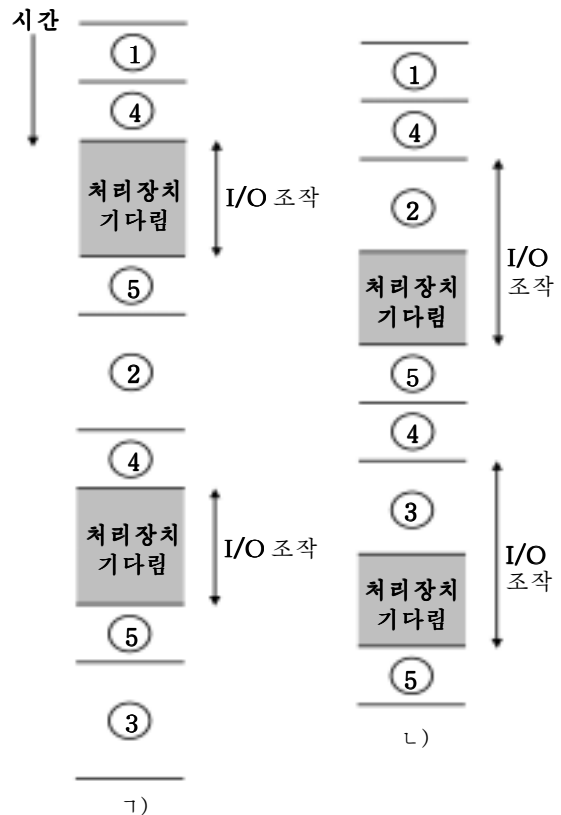



그림 3-11. 프로그람동기 I/O 기다림
ㄱ- 새치기 없음, ㄴ- 새치기 있음

신규약에 의존하는 한개의 문자 혹은 블로크이다. 어떤 경우에는 인쇄기처리새치기가 처리되는 동안 통신새치기가 발생할수 있어야 한다.

다중새치기에 대하여 두가지 방법이 주어 질수 있다. 첫째로는 하나의 새치기가 처리되고 있는동안 새치기들을 리용할수 없게 하는것이다. 이 금지형새치기는 처리장치가

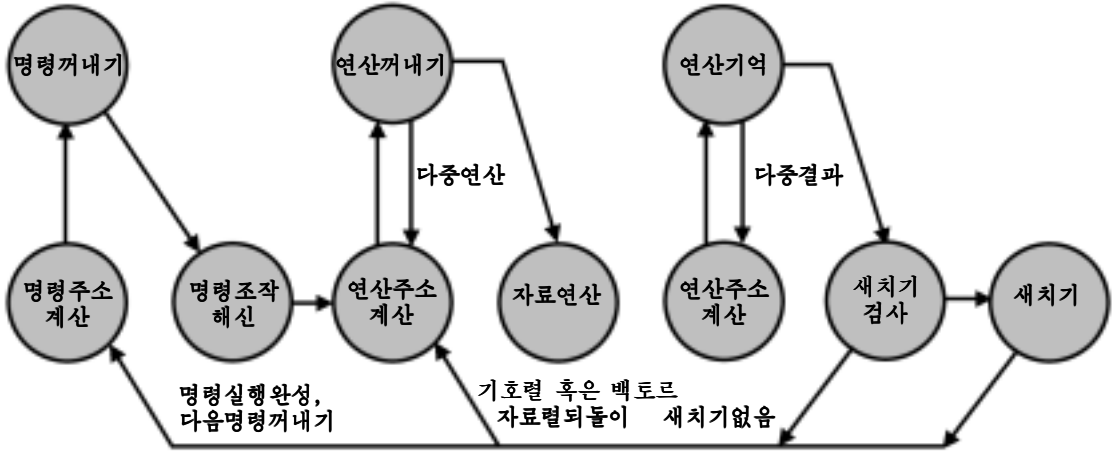



그림 3-12. 새치기 있는 경우 명령주기상태도

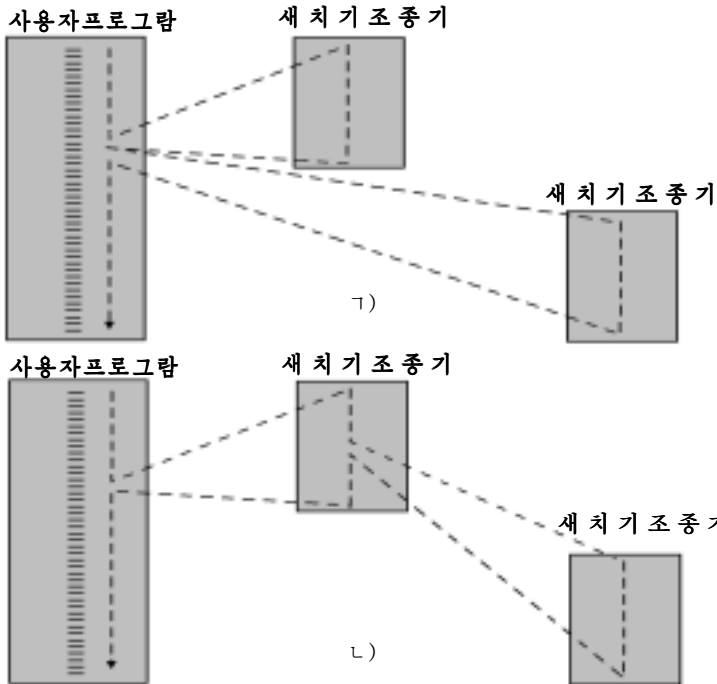



그림 3-13. 다중새치기의 조종흐름
ㄱ-순차적인 새치기처리, ㄴ-우선권새치기처리

새치기요구신호를 무시하거나 접수할수 있게 하는 단순한 방법이다. 만일 새치기가 금지되여 있는 동안에 발생하였다면 일반적으로 그 새치기는 대기상태에 있어야 하며 처리장치가 새치기를 허가한후에 처리장치에 의하여 검사되여야 한다. 따라서 사용자프로그람이 집행되고 있고 새치기가 발생할 때 새치기는 즉시 금지된다. 새치기조종프로그람이 완성된후에 새치기는 사용자프로그람을 다시 집행하기전에 허가되며 처리장치는 추가적인 새치기들이 발생하였는가를 검사한다. 이 방법은 새치기가 엄격한 순서에 따라 조종되게 하는 간단한 방법이다(그림 3-13 ㄱ).

앞의 방법의 결함은 상대적으로 앞선처리 또는 가장 긴급한 처리요구를 실현할수 없는것이다. 실례로 자료가 통신선으로부터 입력되였을 때 많은 그 자료들을 보관하기 위한 처리장소를 확보하기 위한 처리를 고속으로 진행하여

야 한다. 만일 첫번째 입력묶음자료가 두번째 묶음자료가 들어 올 때까지 처리하지 못하면 잃어 버리게 된다.

두번째 방법은 새치기들에 대하여 우선권을 정의하고 낮은 우선권을 가진 새치기가 처리되고 있는 경우에 더 높은 우선권을 가진 새치기를 처리하도록 허락하는 방법이다(그림 3-13 ㄴ). 이 두번째 방법의 실례로서 세개의 I/O 장치들인 인쇄기, 디스크, 통신선들로 이루어 진 체계를 고찰하자. 이 세개의 장치는 우선권이 2, 4, 5 로 주어 졌다. 그림 3-14 는 가능한 처리순서를 보여 준다. 사용자프로그람은 t=0 에서 시작한다. t=10 에서 인쇄기새치기가 발생하며 사용자프로그람정보는 체계탄창에 보관되고 집행은 인쇄기새치기봉사프로그람에로 넘어 간다. 이 처리프로그람이 계속 집행되고 있을 때 t=15 에서 통신새치기봉사프로그람(ISR)에로 넘어 간다. 이 봉사프로그람이 집행되고 있을 때 t=20 에서 디스크새치기가 발생한다. 이 새치기는 우선권이 낮으므로 당분간 유지되고 통신 ISR 의 집행을 완료한다.

통신 ISR 가 t=25 에서 완료되면 이미전의 처리장치상태가 인쇄기 ISR 의 집행상태로 회복된다. 그러나 이 프로그람의 한개 명령이 실행되기전에 처리장치는 더 우선권이 높은 디스크새치기를 접수하고 디스크 ISR 에로 조종을 이행한다. 이 처리프로그람이 t=35 에서 집행이 끝났을 때 인쇄기 ISR 가 다시 회복된다. 인쇄기 ISR 가 t=40 에서 집행이 끝나면 사용자프로그람에로 되돌아 간다.

3. I/O 기능

지금까지는 처리장치에 의하여 조종되는 콤퓨터연산에 대하여 론의하였으며 처리장치와 기억기사이의 호상작용에 대하여 기본적으로 고찰하였다. 여기서는 I/O 구성요소의 역할에 대하여서만 본다. 이 역할은 제 6 장에서 상세히 고찰하므로 여기서는 간단히 개괄한다.

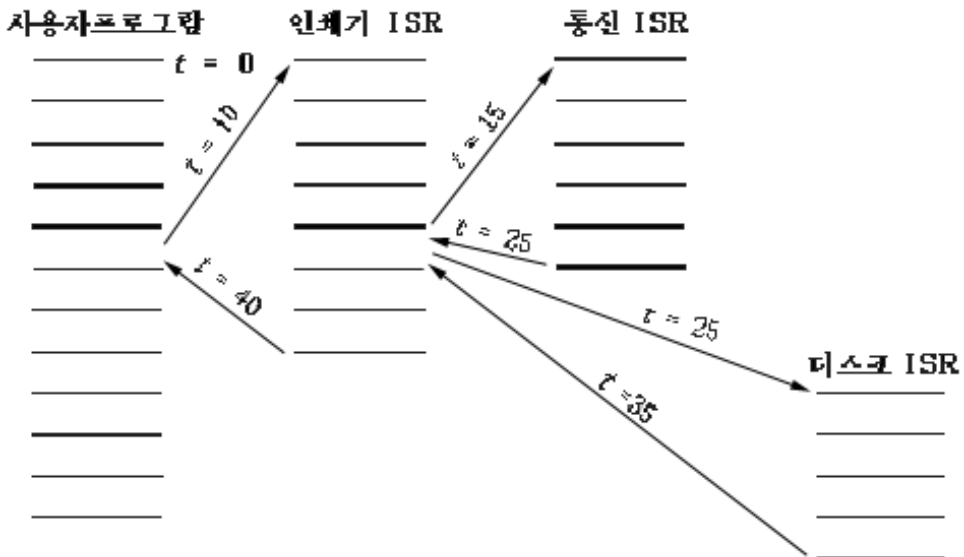



그림 3-14. 다중새치기의 동기순서실례

어떤 I/O 장치(실례로 디스크조종기)는 직접 처리장치와 자료를 교환할수 있다. 즉 처리장치는 기억기와 읽기 혹은 쓰기를 진행할수 있지만 특정의 위치주소를 지적하여 처리장치는 I/O 장치로부터 자료를 읽거나 I/O 장치에 자료쓰기를 할수 있다. 이 경우에

처리장치는 특별한 I/O 장치에 의하여 조종되는 주어 진 장치를 지적한다. 따라서 명령집행순서는 그림 3-5 의 형태와 비슷하며 기억참조명령대신에 I/O 명령을 사용할뿐이다.

일부 경우에는 I/O 가 기억기와 직접 자료를 교환할것을 요구한다. 이 경우에 I/O 와 기억기사이의 자료전송이 처리장치가 없이 진행할수 있게 처리장치는 I/O 장치에 기억장치에 대한 읽기, 쓰기할수 있는 권한을 준다. 이렇게 전송하는동안 I/O 장치가 기억장치에 읽기 혹은 쓰기지령을 주며 자료교환은 위한 처리장치의 역할을 덜어 준다. 이 조작은 직접기억접근(DMA)라고 하며 제 6 장에서 설명한다.

제 3 절. 호상접속구조

콤퓨터는 구성요소들인 세개의 기본장치(처리장치, 기억기, I/O 장치)들의 모임으로 구성된다. 실제적으로 콤퓨터는 기본장치들의 접속망이다. 따라서 장치들을 련결하는 통로가 있어야 한다.

여러개의 장치를 련결하는 통로전체를 **호상접속구조**라고 한다. 이 구조의 설계는 장치들사이에 이루어 져야 하는 교환들에 의존한다.

그림 3-15 는 매개 장치에서 입구와 출구의 기본형태들을 지적하는데 요구되는 교환들의 형태들을 주었다.

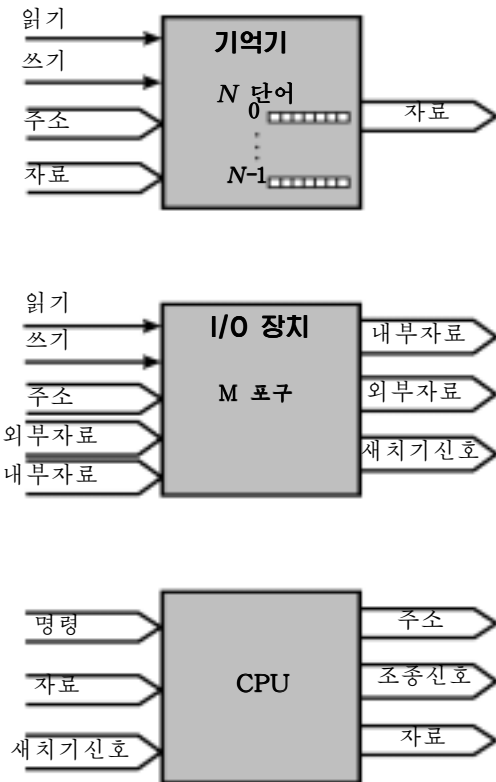



그림 3-15. 콤퓨터의 장치들

- **기억기:** 일반적으로 기억기는 같은 길이를 가지는 N 개의 단어들로 이루어 진다. 매 단어는 유일번호주소(0, 1, …, N-1)로 배렬된다. 자료단어는 기억기에서 읽어 지거나 기억기에 써넣어 진다. 이 조작은 읽기 혹은 쓰기조종신호에 의하여 조종된다. 읽기, 쓰기위치는 주소에 의하여 규정된다.
- **I/O 장치:** 콤퓨터체계의 견지에서 보면 I/O는 기억기와 류사하다. 즉 읽기 혹은 쓰기 두가지 조작을 진행할수 있다. 앞으로 I/O장치는 여러개의 외부장치들을 조종한다. 포구를 통하여 외부장치에 대한 접속을 진행할수 있으며 포구는 유일주소(0, 1, …, M-1)로 주어 진다. 또한 외부장치에 대한 자료의 입력과 출력을 위한 외부자료통로들이 있다. 끝으로 I/O 장치는 처리장치에 새치기조종신호를 보낼수 있어야 한다.
- **처리장치:** 처리장치는 처리를 위한 명령과 자료의 읽기, 처리후의 자료쓰기를 진행하며 체계의 전반적인 조작

을 위한 조종신호들을 내보낸다. 또한 새치기신호를 접수한다.

우의 내용은 자료교환을 정의하였다. 호상접속구조는 다음과 같은 전송형태들을 지원하여야 한다.

- **기억기로부터 처리장치에로 동작:** 처리장치는 기억기로부터 명령과 자료단위를 읽는다.
- **처리장치로부터 기억기에로의 동작:** 처리장치는 기억기에 자료단위를 쓴다.
- **I/O 로부터 처리장치에로의 동작:** 처리장치는 I/O 장치로부터 I/O 장치를 거쳐 자료를 읽는다.
- **처리장치로부터 I/O 장치에로의 동작:** 처리장치는 I/O 장치에 자료를 보낸다.
- **I/O 장치와 기억기사이의 동작:** I/O 장치로부터 기억기 그리고 기억기로부터 I/O 장치에로의 두가지 동작에 대하여 I/O 장치는 처리장치를 거치지 않고 DMA 를 리용하여 기억기와 직접 자료교환을 허락한다.

몇년 지나서 여러개의 호상접속구조들이 나올것이다. 오랜기간에 걸쳐 가장 공통적인것은 모선과 여러개의 다중모선구조이다. 이 장의 뒤부분에서는 모선구조들을 평가하였다.

제 4 절. 모선호상접속

모선은 두개이상의 장치들을 련결하는 통신로이다. 모선의 기본특징은 공유된 전송매체이다. 여러개의 장치들은 모선에 련결되며 임의의 어떤 장치에 의하여 전송된 신호는 모선에 접속된 다른 모든 장치들에 수신된다. 만일 두개의 장치들이 동시에 전송한다면 그 신호들은 겹치며 혼합된다. 따라서 오직 한순간에 한개의 장치만이 성과적으로 전송할수 있다.

일반적으로 모선은 여러개의 통신로들 혹은 선들로 구성된다. 매개 선들은 2 진수 1 과 0 으로 표시되는 신호를 전송할수 있다. 오랜기간 2 진수자들을 전송하는데 하나의 선이 리용되였다. 실례로 자료의 8bit 단위는 8 개선으로 전송될수 있다.

콤퓨터체계들은 콤퓨터체계계층의 여러 준위들에서 구성요소들사이에 경로를 제공하는 여러가지 서로 다른 모선들을 가진다. 기본콤퓨터구성요소들을 련결하는 모선을 **체계모선**이라고 한다. 대체로 일반콤퓨터의 호상접속구조는 하나이상의 체계모선의 리용하는데에 기초한다.

1. 모선구조

체계모선은 일반적으로 50~100 개까지의 분할된 선들로 구성된다. 매개 선들에는 독립적인 의미 혹은 기능이 할당된다. 비록 여러개의 서로 다른 모선들이 있다고 하더라도 임의의 모선에 있는 선들은 세개의 기능그룹(그림 3-16)으로 분류된다. 즉 자료, 주소, 조종모선들이다. 추가적으로 체계안의 모듈들에 전원을 공급하는 전원선들이 있어야 한다.

자료모선들은 체계모듈들사이에 자료를 전송하는 경로를 제공한다. 정확히 말하면

이 선들을 **자료모선**이라고 한다. 일반적으로 자료모선은 **자료모선폭**이라고 하는 8, 16, 32 개의 분리된 선들로 이루어 진다. 매선은 한번에 1bit 를 전송하므로 선들의 수는 한번에 얼마만한 비트를 전송하는가를 결정한다. 자료모선의 폭은 총체적인 체계성능의 기본요인이다. 실례로 자료모선이 8bit 폭이고 매 명령이 16bit 길이를 가진다면 이때 처리장치는 2 배의 명령주기시간만큼 기억기장치를 접근하여야 한다.

주소모선은 자료모선에서 자료의 원천 혹은 목적을 지적하기 위하여 리용된다. 실례로 만일 처리장치가 기억기로부터의 자료를 읽을것을 바란다면 주소모선에 요구하는 단어의 주소를 내보낸다. 명백히 주소모선의 폭은 체계에서 최대로 가능한 기억기용량을 결정한다. 주소모선들은 일반적으로 I/O 포구주소지정에도 리용된다. 일반적으로 높은 위치비트들은 모선에서 개별적인 기억기를 선택하며 낮은 위치의 비트들은 모듈안에서 기억주소 혹은 I/O 포구를 선택한다. 실례로 8bit 모선에서 주소 01111111 이하는 128 개의 단어를 가진 기억기(모듈 0)에서의 주소들을 지적하고 주소 10000000이상은 I/O장치(모듈 1)에 포함되여 있는 I/O 장치를 지적하게 할수 있다.

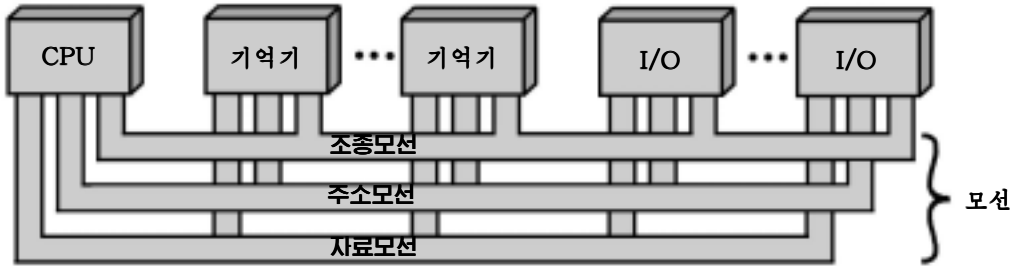



그림 3-16. 모선호상접속방법

조종모선들은 주소와 자료모선들의 접근과 리용을 조종하는데 리용된다. 자료와 주소모선들은 모든 구성요소들에서 공동이므로 그것들의 리용을 조종하는 방법이 있어야 한다. 조종신호들은 체계의 장치들사이에 지령과 동기정보를 전송한다. 동기신호들은 자료와 주소정보의 유효성을 가리킨다. 지령신호들은 수행되는 조작을 규정한다. 대표적인 조종모선들은 다음과 같다.

- **기억기쓰기:** 주소로 지적된 위치에 쓰기 위하여 모선에 자료를 내보내게 한다.
- **기억기읽기:** 주소로 지적된 위치로부터 모선에 자료를 내보내게 한다.
- **I/O 쓰기:** 주소로 지적된 I/O 포구에 출구하기 위하여 모선에 자료를 내보내게 한다.
- **전송응답:** 자료를 모선으로부터 받거나 모선에 내보낸다는것을 가리킨다.
- **모선요구:** 장치가 모선의 조종을 차지할것을 요구한다는것을 가리킨다.
- **모선허가:** 모선을 요구하는 장치가 모선을 조종해도 된다는것을 가리킨다.
- **새치기요구:** 새치기가 요구되였다는것을 가리킨다.
- **새치기응답:** 새치기요구가 접수되였다는것을 알린다.
- **동기박자:** 조작을 동기화하기 위하여 리용한다.
- **재설정:** 모든 장치들을 초기화한다.

모선의 조작은 다음과 같다. 어떤 장치가 다른 장치에 자료를 전송할것을 요구한다면 두가지 작업을 해야 한다. 즉 모선리용을 획득하고 모선에 자료를 전송해야 한다.

만일 어떤 장치가 다른 장치로부터 자료를 요구한다면 모선리용을 획득해야 하며 적당한 조종과 주소모선으로 그 장치에 요구를 전송하여야 한다. 그다음 그 장치가 자료를 보내는 동안 기다려야 한다.

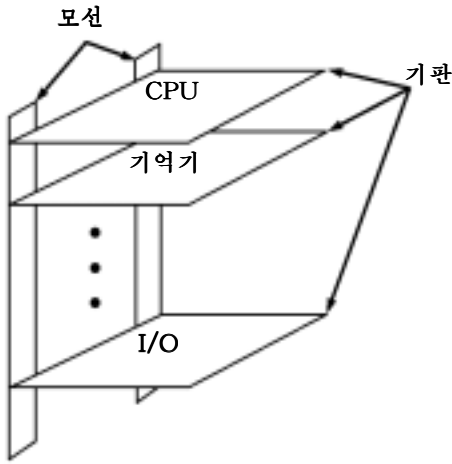



그림 3-17. 모선기본방식의 표준적인 물리적실체

물리적인 체계모선은 여러개의 병렬전기도선이다. 이 도선들은 기판 혹은 인쇄기판에 부식된 금속선들이다. 모선은 모든 체계구성요소에로 확장된다. 가장 일반적인 물리적구성들을 그림 3-17 에 주었다. 이 실례에서 모선은 두개의 수직인 도선들의 렬로 이루어 졌다. 이 수직렬을 따라 규칙적인 간격으로 인쇄기판을 끼워 넣기 위한 수평방향으로 확장홈형식으로 된 접속점들이 있다. 기본체계구성요소들의 매개는 하나이상의 기판들을 가지며 이 홈들에서 모선으로 접속된다. 전체 배렬은 하나의 구조로 묶어진다.

이 배렬은 가장 합리적인것이다. 작은 콤퓨터체계는 이와 같이 되여야 하며 그다음 더 많은 기판(더 많은 기억기, I/O 장치)들을 추가하여야 한다. 만일 기판에서 어떤 구성요소이 고장났다면 그 기판을 쉽게 교체할수 있다.

2. 다중모선계층

여러개의 장치들이 모선에 련결되면 성능은 떨어 질것이다. 여기에는 다음과 같은 두가지 기본원인이 있다.

- 일반적으로 많은 장치들이 모선에 접속되면 모선길이가 길어 지고 결국 더 긴 전송지연시간을 가지게 된다. 이 지연은 장치들이 모선을 리용하도록 조종하는 시간을 결정한다. 모선의 조종이 어떤 장치로부터 다른 장치에로 자주 통과할 때 이 전송지연은 성능에 현저한 영향을 미친다.
- 모선은 전체적인 자료전송요구들이 모선의 최대한 능력을 접근할 때 충돌현상이 발생하게 된다. 이 문제는 모선의 자료전송속도를 높이고 또한 더 넓은 모선(실례로 자료모선이 32 로부터 64bit 로 증가)을 리용함으로써 어느 정도 해결할수 있다. 그러나 장치들(실례로 영상표시장치와 비데오조종기,망접속부)을 접속하면 높은 자료전송속도를 요구하게 되는데 이로하여 경쟁이 일어 나게 된다. 이것은 단일모선으로는 해결할수 없다.

따라서 대부분 콤퓨터체계들은 다중모선을 많이 리용하여야 하며 일반적으로 계층을 이룬다. 전통적으로 표준적인 구조를 그림 3-18 ㄱ에 보여 주었다. 처리장치를 캐쉬기억기에 련결하고 또한 여러개의 국부장치들을 지원하는 국부모선이 있다. 캐쉬기억기조종기는 캐쉬기억기를 이 국부모선뿐아니라 모든 주기억장치에 접속되여 있는 체계모선에 련결한다. 제 4 장에서 론의하는것처럼 캐쉬구조의 리용은 흔히 처리장치를 주기억기접근요구로부터 분리시킨다. 결국 주기억기는 체계모선밖에 있는 국부모선과 분리된다. 이 방법에서 I/O 와 주기억사이의 전송은 능률적으로 CPU 의 참가없이 체계모선을 통과한다.

체계모선밖에 I/O 조종기를 직접 련결할수 있다. 더 능률적인 해결방법은 이 목적을 위하여 여러개의 확장모선을 리용하는것이다. 확장모선대면부는 체계모선과 확장모선에 있는 I/O 조종기들사이에 전송되는 자료를 완충한다. 이 구성은 체계가 매우 다양한 I/O

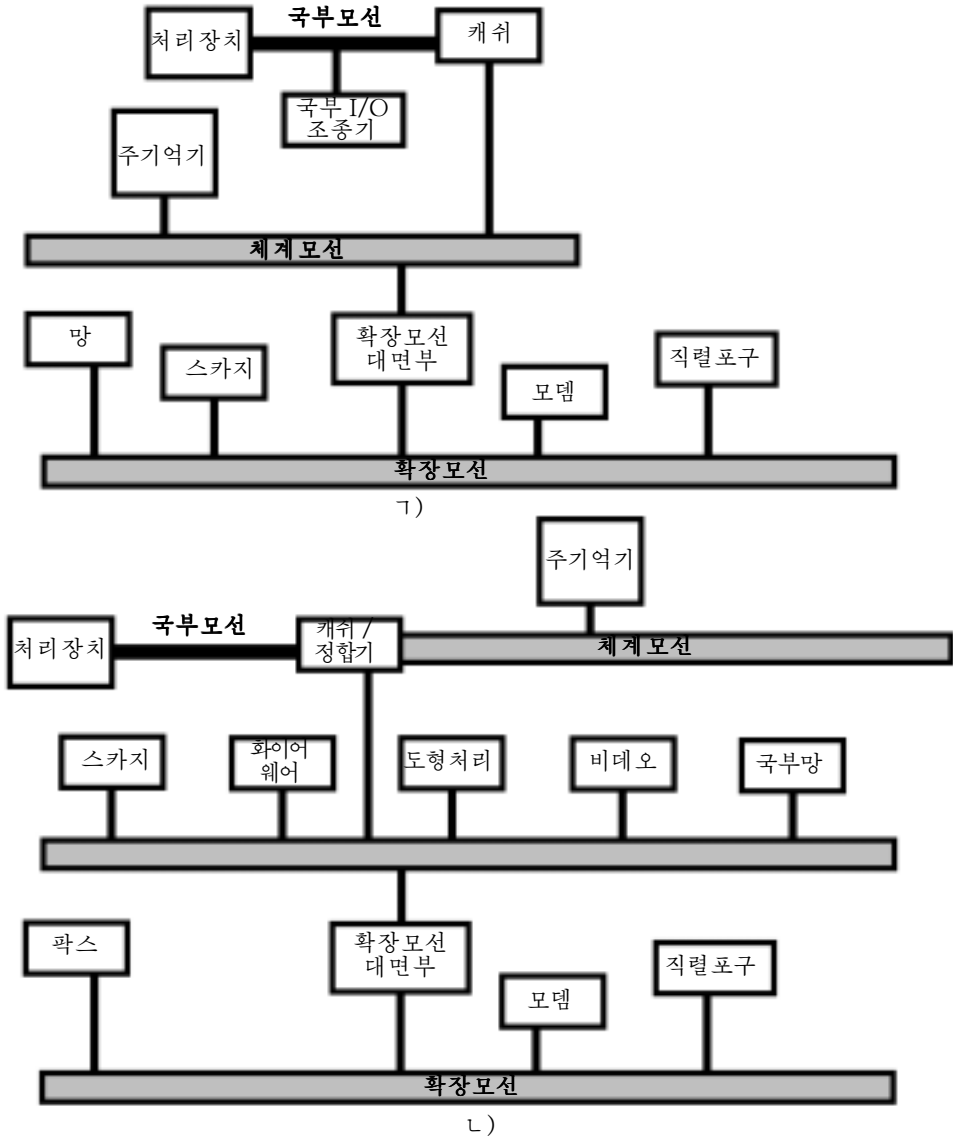



그림 3-18. 모선주기의 실례
ㄱ- 전통적인 모선기본방식, ㄴ- 고성능 모선기본방식

장치들을 접속할수 있게 하며 동시에 I/O 자료전송으로부터 기억기와 처리장치사이의 전송을 분리시킨다.

그림 3-18 ㄱ에 확장모선에 접속되는 몇개의 대표적인 I/O 장치들의 실례를 보여 주었다. 망접속은 10Mbps 모선형망과 같은 국부망, 묶음교환망과 같은 광대역망을 포함한다. SCSI(Small Computer System Interface)는 그자체가 국부디스크장치와 다른 주변

장치들에 리용되는 모선의 한 형식이다. 병렬포구는 인쇄기 혹은 화상입력장치를 접속하는데 리용된다.

이 전통적인 모선방식은 일정한 처리능력을 가지지만 I/O 장치들에서 더 높은 성능을 요구하는 경우에는 불합리하다. 요구들이 증가함에 따라 공업적으로 실현되는 일반방법은 처리장치모선과 고속모선사이의 다리를 요구하며 체계에 상주하면서 조밀하게 묶어 진 고속모선을 구축하는것이다. 이런 배렬은 때때로 중간층 기본방식이라고도 한다.

그림 3-18 ㄴ에 이 방법을 실현한 대표적인 실례를 보여 주었다. 즉 주기억기를 지원하는 체계모선에 일치하게 련결되는 캐쉬조종기에 처리장치를 련결하는 국부모선이 있다. 이 모선은 100Mbps 의 속도를 가진 고속 LAN(국부망)들과 비데오와 도형전용워크스테이션조종기 그리고 SCSI 와 FireWire(100Mbps 직렬모선)를 포함하는 국부주변모선들에 대한 접속조종기들에 접속된다. 그뒤층은 고성능 I/O 장치들을 접속하기 위하여 특별히 설계된 고속모선배렬이다. 저속주변장치들은 여전히 확장모선과 고속모선들사이에 전송자료들을 완충하는 대면부가 있는 확장모선에 의하여 접속된다.

이 배렬의 우점은 우선 고속모선이 처리장치와 함께 설치되여 있는 주기판에 높은 요구를 제기하는 장치들을 직접 설치할수 있는것이며 다음으로 처리장치의 독립성을 실현할수 있는것이다. 따라서 처리장치와 고속모선에서의 차이점은 속도들과 신호선정의들이 그대로 허용되는것이다. 처리소자의 기본방식의 변화는 고속모선에 영향을 미칠수도 있고 미치지 않을수도 있다.

3. 모선설계의 기본요소

여러가지 모선실현방법들이 존재하지만 모선들을 분류하고 구별하는것을 도와 주는 기본파라메터 혹은 설계요소들이 있다. 표 3-2 에 기본요소들을 보여 주었다.

모선형태

모선의 선들은 두개의 일반형태 즉 전용화된것과 다중화된것으로 나눌수 있다. 전용화된 모선은 하나의 기능으로 혹은 콤퓨터구성요소들의 하나의 물리적보조묶음으로 영구적으로 배치된다.

표 3-2. 모선설계의 요소

형태	**모선너비**
전용화	주소
다중화	자료
중재방법	**자료전송형태**
집중방법	읽기
분산방법	쓰기
	읽기변경쓰기
	읽기후쓰기
	블로크

기능적전용화의 실례는 많은 모선들에서 일반적으로 구성되는 분할된 전용주소모선과 자료모선들의 리용하는 방법이다. 그러나 이것이 기본은 아니다. 실례로 주소와 자료정보가 주소유지조종모선을 리용하여 같은 선묶음으로 전송될수 있다. 자료전송의 시작에서 주소는 모선에 배치되며 주소유지선이 능동으로 된다. 이 시점에서 매 모듈은 주소를 복사하기 위한 지정된 주기시간을 가지며 주소화된 장치를 결정한다. 만일 주소화된 장치가 있다면 그다음 주소는 모선으로부터 분리되며 같은 모선접속은 계속하여 읽기 혹은 쓰기자료전송에 리용된다. 여러가지 같은 모선을 리용하는 이 방법을 **시간다중화방식**이라고 한다.

시간다중화방식의 우점은 적은 모선을 리용하는것이다. 결함은 매 장치에서 더 복잡한 회로들이 요구된다는것이다. 또한 같은 모선들을 공유하는 확실한 사건들이 병렬

로 배치될수 없기때문에 성능이 감소될 가능성이 있다.

물리적전용화를 다중모선의 리용방식이라고 하며 매개 모선들은 오직 보조장치에만 접속된다. 대표적인 실례는 모든 I/O 장치를 서로 련결하는 I/O 모선의 리용이다. 즉 이 모선은 몇가지 형태의 I/O 접속장치를 통하여 주모선에 접속된다. 물리적전용화의 우점은 모선경쟁이 없으므로 높은 처리능력을 가질수 있다는것이다. 결함은 체계의 규모와 가격이 높아 지는것이다.

중재방식

가장 단순한 체계를 포함하여 모든 체계에서 여러개의 장치가 모선의 조종을 요구할수 있다. 실례로 I/O 장치가 중앙처리장치의 참가없이 직접 기억기에 대한 읽기 또는 쓰기를 요구할수 있다. 모선을 통하여 한순간에 하나의 장치만이 충분히 자료를 전송할수 있으므로 일부 중재방식들이 요구된다. 여러가지 방식들은 일반적으로 집중화 혹은 분산화방식으로 분류할수 있다. 집중화방식에서 **모선조종기** 혹은 **중재기**라고 하는 장치는 모선에 대한 시간배정을 담당한다. 이 장치는 분리된 모듈 혹은 처리장치의 부분들에 있어야 한다. 분산화방식에는 중심조종기가 없다. 오히려 매개 장치는 접근조종론리를 가지며 장치들은 모선을 공유하도록 작용한다. 중재의 두 방식은 모두 목적이 주장치로서 처리장치 혹은 I/O 장치중의 어느 한 장치를 가리키는것이다. 그다음 주장치는 자료교환을 위하여 피동요소로서 동작하는 일부 다른 장치들과 자료전송에 착수한다.

동기화

동기화는 사건들이 모선에서 균형을 맞추도록 하는 방법이라고 할수 있다. 동기식동기화인 경우 모선에서 사건들의 발생은 박자에 의하여 결정된다. 모선은 박자가 등간격시간의 1 과 0 들을 번갈아 규칙적인 순서로 전송하는 박자선을 포함한다. 신호 1과 0의 전송을 **박자주기** 혹은 **모선박자**라고 하며 주기라고도 한다. 모선에 접속된 모든 다른 장치들은 박자선을 읽을수 있으며 또한 모든 사건들은 박자주기가 시작하는 위치에서 시작한다. 그림 3-19 ㄱ는 동기식읽기조작(동기선도를 설명한 부록 3-1 참고)을 위한 시간선도를 보여 준다. 다른 모선신호들은 박자신호의 오름면에서 변한다. 대부분의 사건들은 한개의 박자주기를 점유한다. 이 간단한 실례에서 처리장치는 읽기신호를 보내고 주소모선에 기억기주소를 배치한다. 또한 모선에 주소와 조종정보의 유효상태를 지적하기 위한 시작신호를 보낸다. 기억기모듈은 주소에 응답하여 한주기후에 모선에 자료와 응답신호를 보낸다.

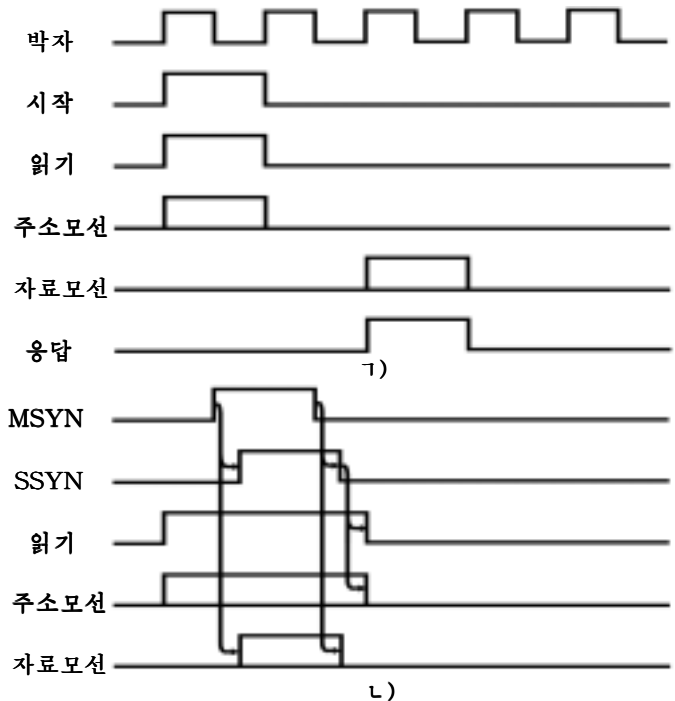



그림 3-19. 읽기조작시간선도
ㄱ- 동기식동기화, ㄴ- 비동기식동기화

비동기식동기화인 경우 모선에서 한사건의 발생은 앞선 사건의 발생에 뒤따르며 의존된다. 그림 3-19 ㄴ의 간단한 실례에서 처리장치는 모선에 주소와 읽기신호를 보낸다. 이 신호들이 안정상태로 유지된후에 확고한 주소와 조종신호들의 유효상태를 가리키는 MSYN(주동기)신호를 내보낸다. 기억기모듈은 자료와 함께 응답을 가리키는 SSYN(피동기)신호로 응답한다. 일단 주장치가 자료모선으로부터 자료를 읽으면 MSYN 신호를 중지시킨다. 이것은 기억기장치가 자료와 SSYN선들을 차단하게 한다. 끝으로 SSYN선이 일단 차단되면 주장치는 읽기신호와 주소정보를 다시 전송한다.

동기식동기화를 실현하고 검사하는것은 간단하다. 그러나 비동기식동기화보다 유연성이 부족하다. 동기식모선에서 모든 장치들은 고정박자속도의 제한을 받으므로 체계는 장치성능이 개선되여도 그 우점이 나타나지 않는다. 비동기화에서서는 낡은 기술로 만들어 진 속도가 뜬 장치와 새로운 기술에 의해 만들어 진 속도가 빠른 장치들을 섞어서 체계를 만들어도 그들 모두가 모선을 공유할수 있다.

모선폭

이미 모선폭에 대한 개념을 고찰하였다. 자료모선의 폭은 체계의 성능에 영향을 준다. 더 넓은 모선은 더 많은 비트들을 한번에 전송한다. 넓은 주소모선은 체계의 기억용량에 영향을 준다. 주소모선이 넓으면 더 넓은 범위의 주소를 참조할수 있다.

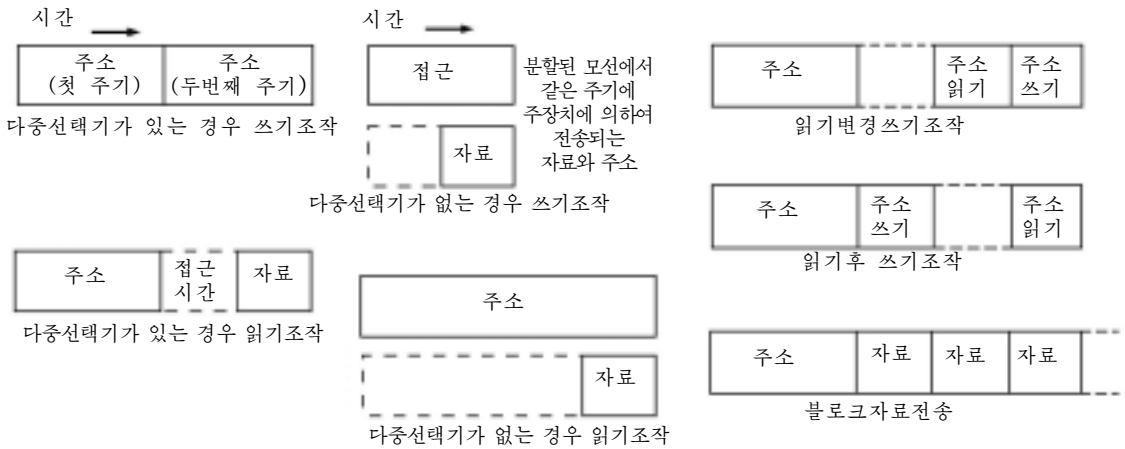



그림 3-20. 모선자료전송형식

자료전송형식

끝으로 모선은 그림 3-20 에서 보여 주는것처럼 여러가지 자료전송형식을 제공한다. 모든 모선들은 쓰기와 읽기전송을 가진다. 다중화된 주소/자료모선의 경우에 모선은 먼저 주소를 지정하고 그다음 자료를 전송하는데 리용된다. 읽기조작에서 모선에 접속되여 있는 피동장치로부터 자료를 꺼내는 동안 일반적으로 기다림을 가진다. 읽기 혹은 쓰기에서 나머지조작을 위한 모선조종권(읽기/쓰기를 요구하기 위한 모선점유와 그후 읽기/쓰기를 진행하기 위한 모선점유)을 얻기 위하여 중재과정을 거쳐야 한다면 역시 지연이 필요하다(실례로 읽기쓰기를 요구하는 모선을 장악한 다음 , 읽기쓰기를 수행하기 위하여 다시 모선을 장악한다.).

주소와 자료모선을 전용화하는 경우에 주소는 주소모선에 놓이며 자료는 자료모선에 놓인다. 쓰기조작에서 주장치는 자료모선에 자료를 출력하고 즉시로 주소를 고정시

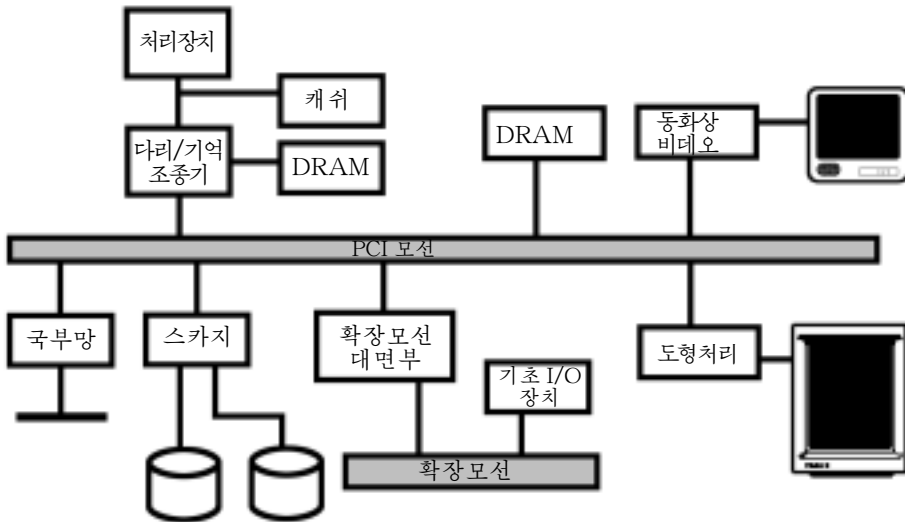
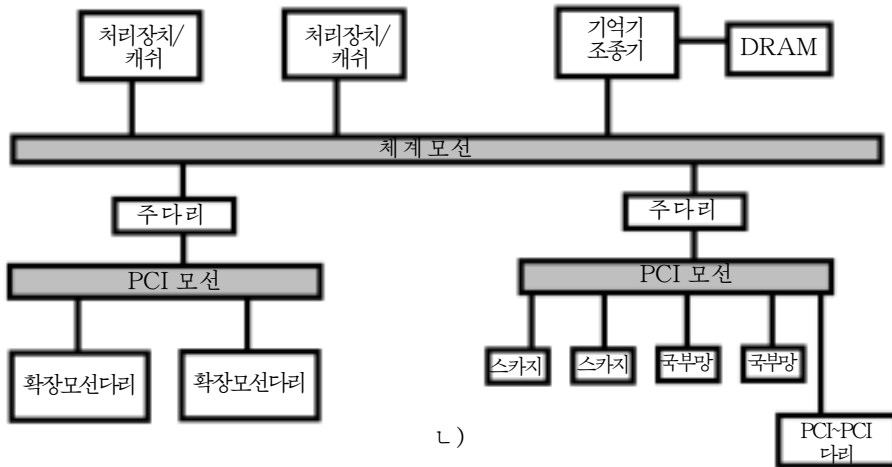



그림 3-21. PCI 구성의 실례

ㄱ- 일반적인 탁상형콤퓨터체계, ㄴ- 일반적인 봉사기체계

킨다. 그리고 보조장치는 그 주소에 대응하여 능동상태로 된다. 읽기조작에서 종속장치는 주소에 대응하여 자료모선에 자료를 내보내며 주장치는 자료를 읽어 들인다.

또한 몇개의 모선들을 허가하는 여러가지 접속조작들이 있다. 읽기변경쓰기(쓰기후읽기)조작은 같은 주소에 대하여 쓰기를 한 다음 즉시에 읽기를 하는 간단한 조작이다. 주소는 이 조작과정에 오직 한번만 나타난다. 다른 모선조종자에 의한 자료접근을 방지하기 위하여 모든 조작은 모두 개별적으로 할수 있다. 이 기능의 기본목적은 다중프로

그람체계에서 공유기억자원을 보호하기 위한것이다(제 7 장 참고).

쓰기후 읽기는 같은 주소로부터 읽기가 직접 뒤따르는 쓰기로 이루어 진 분리할수 없는 조작이다. 이 읽기조작은 검사목적으로 수행된다.

일부 모선체계들은 블로크자료전송을 제공한다. 이 경우에 한개 주소박자에 n 개의 자료박자가 뒤따른다. 첫 자료항목은 지정된 주소에 의하여 전송된다. 나머지자료들은 순서배렬주소에 의하여 전송된다.

제 5 절. PCI

PCI(Peripheral Component Interconnect)는 일반적인 고성능접속부이며 주변모선 혹은 중간층모선으로서의 기능을 수행할수 있는 처리장치와 독립모선이다. 다른 일반모선에 비하여 PCI 는 고속 I/O 보조체계들(실례로 도형표시장치접속기, 망접속조종기, 디스크조종기 등)을 위한 더 좋은 체계성능을 제공한다. 현재 표준은 528Mbyte/s 혹은 4.224Gbps(초당 비트)의 전송속도를 가질수 있는 66MHz 의 64bit 자료모선의 리용을 허가한다. 그러나 PCI 의 매력을 느끼게 하는것은 속도뿐이 아니다. PCI 는 현대체계들의 I/O 요구들을 경제적으로 만족시킬수 있게 특별히 설계되였다. 즉 실현하는데 아주 적은 소편들이 요구되며 PCI 모선에 접속된 다른 모선들을 지원한다.

인텔은 펜티움기준체계에 1990 년부터 PCI 를 설치하였다. 즉시 인텔은 공동소유를 위하여 특허를 공개하였으며 PCI 세부의 호환성을 유지하면서 앞으로 더 발전시키기 위하여 산업련합의 창조물인 PCI SIG 로 발전시켰다. 결과 PCI 는 개인용콤퓨터, 워크스테이션, 봉사기체계들에서 광범히 도입되였으며 더 널리 리용되였다. 여기에서 서술하고 있는것은 그 변종이 1995 년에 공개된 PCI 2.1 이다. 세부가 공개되고 극소형처리장치와 주변장치산업에서 기판상의 부분으로 제공되고 있으므로 여러 판매자들에 의하여 제작된 PCI 는 호환성을 가진다.

PCI 는 단일처리장치체계와 다중처리장치체계들을 포함하는 극소형처리장치기준구성의 다양성을 지원하기 위하여 설계되였다. 따라서 일반목적기능모임들을 제공한다. PCI 는 동기식동기화와 집중화된 중재기설계방안의 리용을 받아 들인것이다.

그림 3-21 ㄱ는 단일처리장치체계에서 PCI 의 리용을 대표적으로 보여 준다. PCI 모선에 접속된 DRAM 조종기와 다리는 처리장치와 련결을 믿음성있게 보장하며 높은 속도의 자료전송을 보장하기 위한 능력을 가진다. 다리는 PCI 모선의 속도가 처리장치의 I/O 속도와 차이나므로 완충기로서 동작한다. 다중처리장치체계(그림 3-21 ㄴ)에서 여러개의 PCI 구성들은 처리장치체계모선에 다리를 통하여 련결된다. 체계모선은 오직 처리장치/캐쉬장치, 주기억기와 PCI 다리들에만 련결된다. 다시말하여 다리의 리용은 처리장치속도와 독립적인 PCI 를 보존하며 여전히 자료수신과 송신을 빠르게 할수 있는 능력을 제공한다.

1. 모선구조

PCI 는 32 혹은 64bit 모선으로 구성되였다. 표 3-3 에서 PCI 에 있는 49 개의 조종신호선들을 정의하였다. 이것들은 다음과 같은 기능그룹들로 나누어 진다.

표 3-3. PCI 지령신호선

신호명	형태	설명
체계단자		
CLK	in	모든 동작에 대한 동기를 제공하며 오름면에서 모든 입력신호들을 접수하게 한다. 33MHz 에 박자속도를 지원한다
RST#	in	초기화상태를 위하여 모든 PCI 특수등록기들, 순서기, 신호들을 초기화한다.
주소와 자료단자		
AD[3::0]	t/s	주소와 자료에 리용되는 다중화된 모선
C/BE[3::0]	t/s	지령과 바이트가능신호를 위한 다중화된 모선. 자료단계에서 이 모선들은 4byte 통로들중에 어느것으로 지적된 단어를 보내겠는가를 가르킨다.
PAR	t/s	한박자후에 AD 와 C/BE 모선들의 기우성을 제공한다. 주장치들은 주소모선과 쓰기자료단계를 위한 PAR 를 구동한다. 대상장치는 읽기자료단계를 위한 PAR 를 구동한다.
대면부조종단자		
FRAME#	s/ t/s	동작의 시작과 진행을 지적하는 현재주장치에 의하여 구동된다. 이 신호는 시작에서 능동으로 되며 초기화장치가 마지막자료단계의 시작을 준비할 때 피동으로 된다.
IRDY#	s/ t/s	초기화장치가 이 신호를 내보낸다. 즉 현재모선주장치(동작의 초기화장치)에 의하여 구동된다. 읽기할 때 주장치가 자료를 접수할 준비가 되였다는것을 지적하며 쓰기할 때 유효자료가 AD 에 존재한다는것을 지적한다.
TRDY#	s/ t/s	대상자가 이 신호를 내보낸다. 즉 대상자(선택된 장치)에 의하여 구동된다. 읽기할 때 유효자료가 AD 에 존재한다는것을 지적하며 쓰기할 때 대상자가 자료를 접수할 준비가 되였다는것을 지적한다.
STOP#	s/ t/s	현재대상자가 초기화장치로 하여금 현재동작을 정지시키도록 요구하는것을 가리킨다.
IDSEL	in	초기화장치선택. 소편이 구성읽기와 쓰기동작을 진행하는것을 선택하는데 리용된다.
DEVSEL#	in	장치선택. 장치의 주소가 인식되였을 때 목적장치를 능동으로 설정한다. 어느 장치가 선택되였는가를 현행초기화장치에게 알려 준다.
중재단자		
REQ#	t/s	장치가 모선을 리용할것을 요구한다는것을 중재기에 알린다. 장치들은 별형방식으로 접속한다.
GNT#	t/s	중재기가 모선접근을 승인한다는것을 장치에 알린다. 장치들은 별형방식으로 접속된다.
오유통보단자		
PERR#	s/t/s	기우성오유. 자료기우성오유가 쓰기단계에서 대상자에 의하여 읽기단계에서는 초기화장치에 의하여 검사되였다는것을 시작한다.
SERR#	o/d	체계오유. 임의의 장치가 기우성오유와 이와 다른 파괴적인 오유들을 통신하려고 할 때 발생한다.

- **체계단자들:** 박자와 재설정단자들을 가진다.
- **주소와 자료단자들:** 주소와 자료들을 시간적으로 분배하는 32 개의 선들이다. 이 그룹의 일부 신호선들은 주소와 자료를 나르는 신호선들을 해석하고 유지하는데 리용된다.

- **대면부조종단자들:** 처리의 동기를 조종하고 초기화장치와 대상자사이의 일치를 보장한다.
- **중재단자들:** 일부 PCI 신호선들과는 달리 이 신호선들은 공유되지 못한다. 오히려 매 PCI 주장치들은 PCI 모선중재기에 직접 련결되는 중재선들과 쌍을 이룬다.
- **오유통보단자들:** 기우성오유와 다른 오유들을 통보하는데 리용된다.

또한 PCI 상세명세표는 아래와 같은 기능그룹들로 구분된 51 개의 선택신호선(표 3-4)들을 정의한다.

표 3-4. PCI 선택적인 신호선

신호명	형태	설명
새치기단자		
INTA#	o/d	새치기를 요구하는데 리용된다.
INTB#	o/d	새치기를 요구하는데 리용된다. 다기능장치에만 해당된다.
INTC#	o/d	새치기를 요구하는데 리용된다. 다기능장치에만 해당된다.
INTD#	o/d	새치기를 요구하는데 리용된다. 다기능장치에만 해당된다.
캐쉬지연단자		
SB	in/out	탐색탈퇴. 변경된 행에 대한 명중을 지적한다.
SDONE	in/out	탐색완료. 현재중점에 대한 탐색의 상태를 가리킨다. 탐색이 완료되였을 때 능동으로 된다.
64bit 모선확장단자		
AD[63::32]	t/s	64bit 모선으로 확장하도록 주소와 자료에 대하여 리용되는 다중화모선
C/BE[7::4]#	t/s	모선지령과 바이트허가신호도 다중화된다. 주소단계에서 이 신호선들은 추가적인 모선지령을 제공한다. 자료단계에서는 4byte 통로들중의 어느 통로로 자료를 전송하는가를 지적한다.
REQ64	s/t/s	64bit 전송을 요구하는데 리용한다.
ACK64#	s/t/s	대상절차. 64bit 전송을 수행하도록 할것을 지시하고 있음을 가리킨다.
PAR64	t/s	한박자 다음에 한박자 AD 와 C/BE 선들에 대한 우수기우성을 제공한다.
JTAG/한계주사단자		
TCK	in	박자검사. 박자상태정보와 경계주사하는 동안 장치에 대한 자료의 입출력검사에 리용된다.
TDI	in	입력검사. 장치에 검사자료와 명령을 련속적으로 밀어 넣는데 리용된다.
TDO	out	출력검사. 장치밖으로 검사자료와 명령들을 련속적으로 밀어 내는데 리용된다.
TMS	in	방식선택검사. 검사접근포구조종기의 상태를 조종하는데 있다.
TRST#	in	재설정검사. 검사접근포구조종기를 초기화하는데 리용된다.

In — 입력신호
Out — 출력신호
t/s — 쌍방향, 3 상태, 입출력신호
s/t/s — 한번에 하나의 소자에 의해서만 구동되는 3 상태신호를 확인
o/d — 열린배출구: 여러 장치들이 배선론리합을 리용하여 공유하자는데 있다.
— 신호의 능동상태가 L 준위임을 표시

- **새치기단자들:** 이 단자들은 봉사요구를 발생해야 하는 PCI 장치들을 위하여 제공한다. 이것들은 중재단자들에 의하여 공유된 선들이 아니다. 오히려 매 PCI 장치들은 새치기조종기들에 자기의 새치기선 혹은 선들을 가진다.
- **캐쉬지원단자들:** 이 단자들은 처리장치 혹은 다른 장치에 림시 완충될수 있는 PCI 의 기억기를 지원하는데 필요하다. 이 단자들은 전용캐슈통신규약들을 지원한다(이와 같은 규약의 론의는 제 16 장 참고).
- **64bit 모선확장단자들:** 주소와 자료들에 시간적으로 다중분배되며 64bit 주소/자료모선형식으로 확장주소/자료모선들로 접속되는 32bit 모선을 포함한다. 이 그룹에서 일부 선들은 주소와 자료모선들을 나르는 신호선들을 해석하고 유지하는데 리용된다. 끝으로 두개의 PCI 장치가 64bit 로 리용될수 있게 하는 두개의 선이 있다.
- **JTAG/경계선 주사단자들:** 이 신호선들은 IEEE 규격 1149.1 에서 정의된 절차검사를 지원한다.

2. PCI 지령

모선활용은 봉사요구자 혹은 모선조종장치와 목적대상자사이의 작용형태로 발생한다. 모선조종장치가 모선의 조종을 가졌을 때 다음에 발생할수 있는 동작의 형태를 결정한다. 작용에서 주소가 모선을 차지하는 동안 C/B 모선들이 동작형태를 통보하는데 리용된다. 지령들에는 다음과 같은것들이 있다.

- 새치기응답
- 특수주기
- I/O 읽기
- I/O 쓰기
- 기억기읽기
- 기억기읽기선
- 기억기읽기다중화
- 기억기쓰기
- 기억기쓰기와 무시
- 구성읽기
- 구성쓰기
- 2 중주소주기

새치기응답은 PCI 모선에서 새치기조종기로서의 기능을 가진 장치에 대한 고위적인 읽기지령이다. 주소모선들은 주소단계동안 리용되지 못하며 바이트허가선들은 되돌려지는 새치기지적자의 크기를 지적한다.

특수주기지령은 초기화장치로 하여금 여러개의 대상자들에 통보를 전면전송하도록 하는데 리용된다.

I/O 읽기쓰기지령은 초기화장치와 I/O 조종기사이의 자료를 전송하는데 리용된다. 매개 I/O 장치는 자기의 I/O 주소공간을 가지며 주소모선들은 개별적인 장치들을 지적하고 그 장치와 전송하는 자료를 지적하는데 리용된다. I/O 주소의 개념은 제 6 장에서 설명하였다.

기억기읽기와 쓰기지령은 자료전송을 지적하는데 리용되며 자료전송은 하나이상의 박자를 가진다. 이 지령들에 대한 해석은 PCI 모선에 있는 기억기조종기가 기억기와 캐쉬사이의 전송을 위한 PCI 통신규약을 지원하는가 하지 않는가에 의존한다. 만일 있다면

기억기와의 자료전송은 일반적으로 캐쉬행들 혹은 블로크들로 진행된다. 3 개의 기억기읽기지령은 표 3-5 에서와 같이 대략적으로 리용된다. 기억기쓰기지령은 기억기에 하나

표 3-5. PCI 읽기지령의 해석

읽기명령형태	캐쉬가능기억기	캐쉬불가능기억기
기억기읽기	한행의 절반이하채우기	두개의 자료전송주기이하채우기
기억기읽기행	3 개의 캐쉬행까지 한행의 절반이상채우기	3∼12 개 자료전송채우기
기억기다중읽기	3개의 캐쉬행이상채우기	12 개 자료이상채우기

이상의 자료주기로 자료전송을 위하여 리용된다. 기억기쓰기와 무시지령은 기억기에 하나이상의 박자들로 자료를 전송한다. 추가적으로 적어도 하나의 캐쉬행을 쓰는것으로 한다. 이 지령은 기억기에 한행을 비동시쓰기하는 캐쉬기능을 지원한다.

주장치는 두개의 설치지령을 통하여 PCI 에 련결된 장치의 설치파라메터를 읽거나 갱신할수 있다. 매 PCI 소자는 256 개까지의 내부등록기를 가지고 있는데 이 등록기들이 체계를 초기화하는 동안 그 소자를 구축하는데 리용된다.

2 중설치주소주기지령은 초기화장치에게 이 지령이 64bit 주소화를 리용하고 있다는것을 알려 주는데 리용된다.

3. 자료전송

PCI 모선에서 모든 자료전송은 하나의 주소단계와 하나이상의 자료단계로 이루어 지는 단일동작이다. 이 론의에서는 표준적인 읽기조작을 설명한다. 쓰기조작은 이와 비슷하다.

그림 3-22 는 읽기조작의 시간선도를 보여 준다. 모든 사건은 매 박자주기의 중간에 있는 박자의 내림면시동으로 동기화된다. 그러나 장치들은 모선주기의 시작오름면에서 모선들을 접수한다. 그림에 표시된 다음과 같은 사건들을 고찰하자.

ㄱ. 일단 모선주장치가 모선의 조종을 획득하면 FRAME 을 능동상태로 만듦으로써 기동을 시작한다. 이 선은 초기화장치가 마지막자료단계를 완성하는것을 준비할 때까지 능동상태를 유지한다. 초기화장치는 또한 주소모선에 시작주소를 전송하며 C/BE 선들에는 읽기지령이 전송된다.

ㄴ. 두번째 박자의 시작에서 대상자는 AD 선들에서 그의 주소에 응답하여야 한다.

ㄷ. 초기화장치는 AD 모선구동을 끝낸다. 련속되는 주기는 하나이상의 장치에 의하여 구동되는 모든 신호선들에 요구된다. 따라서 주소신호의 동작을 끝냄으로써 목적장치들이 모선을 리용하도록 하게 한다. 초기화장치는 현재주소화된 자료(1~4byte)를 전송하는데 AD 모선이 리용되도록 하기 위하여 C/BE 선들의 정보를 변화시킨다. 초기화장치는 또한 첫 자료항목에 대한 준비가 되였다는것을 지시하는 TRDY 신호를 능동으로 설정한다.

ㄹ. 선택된 목적대상자는 그 주소를 인식하고 응답하였다는것은 지적하는 DEVSEL 신호선을 능동으로 설정한다. 대상자는 AD 선들에 요구된 자료를 배치하고 모선에 유효자료가 존재한다는것을 지적하는 TRDY 신호선을 능동으로 한다.

ㅁ. 초기화장치는 박자 4 의 시작점에서 자료를 읽으며 다음읽기에 대한 준비를 요구할 때 바이트허가선들을 변화시킨다.

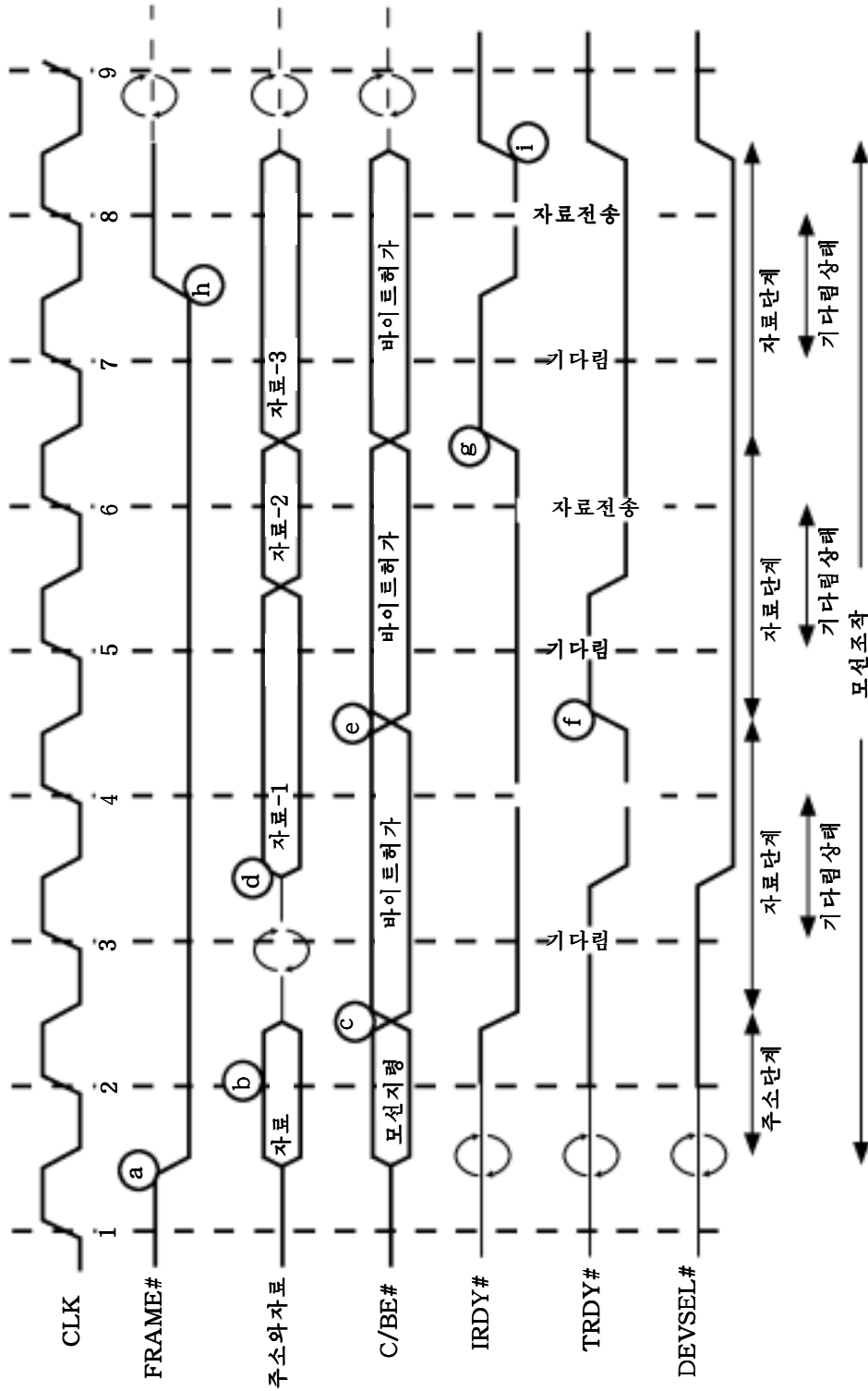



그림 3-22. PCI 읽기조작

ㅂ. 이 실례에서 대상자는 자료전송의 두번째 블로크를 준비하는데 일정한 시간을 요구한다. 그러므로 대상자는 다음주기동안 새로운 자료를 가질수 없는다는것을 초기화장치에 신호하기 위하여 TRDY 신호선을 피동으로 설정한다. 따라서 초기화장치는 5 번째 주기의 시작에서 자료모선을 읽을수 없으며 그 주기동안 바이트허가를 변화시키지 못한다. 자료항목을 읽기 위한 준비를 여전히 하지 못한다. 그러므로 IRDY 신호선이 피동으로 설정된다. 이것은 대상자가 일정한 박자주기동안 모선상에 세번째 자료항을 유지하도록 한다.

ㅅ. 초기화장치는 세번째 자료전송이 마지막이라는것을 알고 따라서 이것이 마지막자료전송이라는것을 대상자에게 신호하기 위하여 FRAME 신호선을 피동으로 만든다. 또한 초기화장치는 이 전송의 완료를 준비한다는것을 신호하기 위하여 IRDY 신호선을 능동으로 설정한다.

ㅇ. 모선을 휴식상태로 되돌리도록 초기화장치는 IRDY 신호선을 피동으로 하며 대상자는 TRDY 와 DEVSEL 신호선을 피동으로 설정한다.

4. 중재

PCI 는 매개 주장치들이 유일한 요구(R 와 허가신호)를 가지는 집중화된 동기식중재방법을 리용한다. 신호선들은 중심중재기에서 형성되며 단순한 요구허가맞잡이조종은 모선에 대한 접근을 확인하는데 리용된다(그림 3-23).

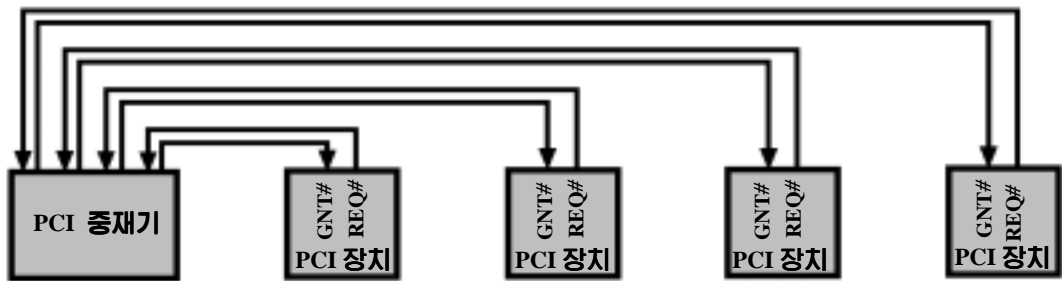



그림 3-23. PCI 모선중재기

PCI세부는 개별적인 중재알고리듬을 지적하지 못한다. 중재기는 선입선출봉사방법, 회전순회방법 혹은 어느 정도의 우선순위방법을 리용할수 있다.

PCI주장치는 하나의 처리가 하나이상의 련속적인 자료단계들과 그에 대응하는 하나의 주소단계로 구성된 동작을 수행하도록 매개 처리에 대하여 중재를 해야 한다.

그림 3-24 는 장치 A 와 B 가 모선에 대하여 중재하고 있는 실례이다.

다음과 같은 순서로 진행된다.

ㄱ. 박자 1 시작의 어떤 앞의 위치에서 A 는 REQ 신호를 내보낸다. 중재기는 박자주기 1 의 시작에서 이 신호를 접수한다.

ㄴ. 박자주기 1 동안 B 는 REQ 신호를 보내여 모선의 리용을 요구한다.

ㄷ. 동시에 중재기는 A 에 대한 모선접근을 허가하도록 GNT-A 를 능동으로 만든다.

ㄹ. 모선조종기 A 는 박자 2 의 시작에서 GNT-A 를 접수하며 모선접근을 하게

한다. 모선이 휴식상태라는것을 지적하는 IRDY 신호선과 TRDY 신호선이 비능동으로 되였는가를 확인한다. 따라서 FRAME 신호선을 능동으로 만들고 주소모선에 주소정보와 C/BE 모선에 지령을 배치한다. 또한 이 동작후에 수행되는 두번째 동작을 하도록 REQ-A 를 계속 능동으로 한다.

ㅁ. 모선중재기는 박자 3 의 시작에서 모든 GNT 모선들을 접수하며 다음동작을 위하여 B 에 모선을 허가하도록 중재결정을 진행한다. 그다음 GNT-B 를 능동으로 하며 GNT-A 를 비능동으로 만든다. B 는 모선의 휴식상태로 되돌아올 때까지 모선을 리용할수 없다.

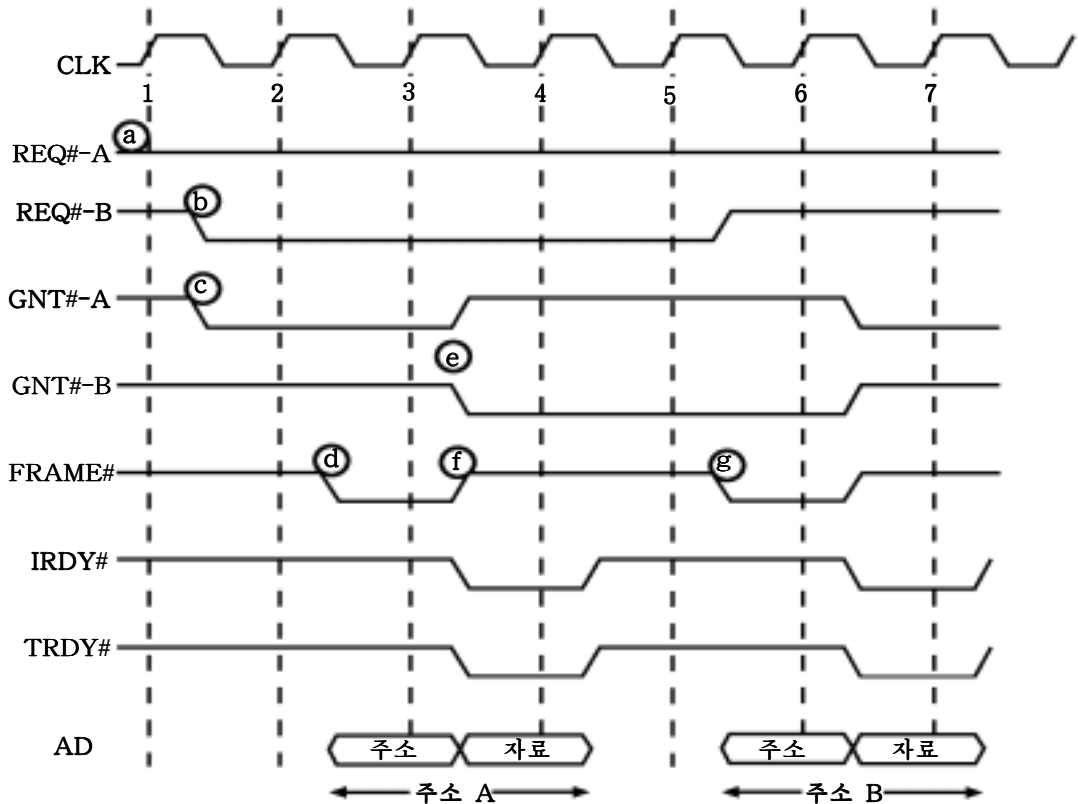



그림 3-24. 두 주장치사이의 PCI 모선중재원리

ㅂ. A 는 마지막자료전송이 진행중이라는것을 지적하도록 FRAME 신호선을 비능동으로 만든다. A 는 자료모선에 자료를 전송하며 IRDY 신호선으로 대상자에게 신호한다. 대상자는 다음박자주기의 시작에서 자료를 읽는다.

ㅅ. 박자 5 의 시작에서 B 는 IRDY 신호선과 FRAME 신호선이 비능동으로 되였다는것을 확인하고 FRAME 신호선을 능동상태로 만들어 모선의 조종을 가질수 있게 한다. 또한 하나의 동작만을 수행할것을 원하므로 REQ 신호선을 비능동으로 만든다.

계속하여 조종기 A 는 다음동작을 위한 모선의 접근을 승인한다.

중재는 현재모선조종기가 자료전송을 수행하고 있는것과 동시에 진행할수 있다는것을 알수 있다. 그러므로 모선주기가 중재수행에서 끊어 지지 않는다.

참고문헌과 Web 싸이트

모선들과 다른 호상접속구조들에 대한 문헌은 그리 많지 못하다. [ALEX93]에서는 여러가지 특수한 모선들의 평가를 포함하여 모선구조와 모선전송지표들에 대하여 심도있게 취급하였다.

PCI 에 대하여 충분하고 명백하게 서술한 책은 [SHAN95], [SOLA94]들이며 여기서는 PCI 에 대한 확고한 많은 정보들을 취급하였다.

Web 싸이트

- PCI special Interest Group: 이 Web 싸이트는 PCI 규정과 상품에 대한 정보를 제공한다.

련습문제

1. 그림 3-4 의 가상적인 처리장치가 다음의 두개의 I/O 명령을 가진다.

   0011 = I/O 로부터 AC 에 넣기
   0111 = I/O 에로 AC 를 출구

   이 경우에 12bit 주소는 개별적인 I/O 장치를 지적한다. 다음의 프로그람에 대한 프로그람집행을 고찰하시오.

   - 장치 5 로부터 AC 에 넣기
   - 기억기위치 940 의 내용과 더하기
   - 장치 6 에로 AC 를 출구

   장치 5 로부터 받은 값은 3 이고 주소 940 은 2 라는 값을 가진다고 가정하시오.

2. 두개의 마당으로 이루어 진 32bit 명령을 가지는 가상적인 32bit 극소형처리장치에 대하여 고찰하자. 여기서 명령의 첫번째 바이트는 조작코드이고 나머지는 직접값연산수 혹은 연산수주소이다.

   ㄱ. 최대로 직접주소화할수 있는 기억기용량(바이트)은 얼마인가?
   ㄴ. 만일 극소형처리장치모선이
   ① 32bit 국부주소모선과 16bit 국부자료모선 혹은
   ② 16bit 국부주소모선과 16bit 국부자료모선일 때 체계속도에 대한 영향을 론의하시오.
   ㄷ. 프로그람계수기와 명령등록기가 얼마만한 비트길이를 요구하는가?
   [ALEX93] 참고

3. 16bit 주소(실례로 프로그람계수기와 주소등록기가 16bit 폭이라고 가정)와 16bit 자료모선을 가지는 가상적인 극소형처리장치를 고찰하자.

   ㄱ.처리장치가 16bit 기억기에 련결될 때 직접 접근할수 있는 최대기억주소공간은 얼마인가?

ㄴ. 처리장치가 8bit 기억기에 련결될 때 직접 접근할수 있는 최대기억주소 공간은 얼마인가?

ㄷ. 개별적인 I/O 공간을 접근하는데 이 처리장치는 어떤 방식적기능을 가질 수 있는가?

ㄹ. I/O 명령이 8bit I/O 포구번호를 지적한다면 극소형처리장치는 얼마나 많은 8bit I/O 포구를 제공할수 있는가, 얼마나 많은 16bit I/O 포구를 제공할수 있는가를 설명하시오.
[ALEX 93] 참고

4. 16bit 의 외부자료모선과 8MHz 로 동작하는 32bit 극소형처리장치를 고찰하자. 이 극소형처리장치는 최소지속시간이 4 개 박자주기인 모선주기를 가진다고 가정하자. 이 극소형처리장치가 유지할수 있는 최대자료전송속도는 얼마인가? 성능을 높이기 위하여 외부자료모선을 32bit 로 만들며 극소형처리장치에 공급되는 외부박자주파수를 2 배로 할수 있는가? 자기가 만든 임의의 다른 가설들을 말하고 설명하시오.
[ALEX93] 참고

5. 간단한 건반/인쇄기를 가진 전신타자기를 조종하는 I/O 장치를 포함하는 콤퓨터 체계를 고찰하자. 다음의 등록기들은 처리장치안에 있으며 체계모선에 직접 련결된다.

INPR: 입구등록기, 8bit
OUTR: 출구등록기, 8bit
FGI: 입력기발, 1bit
FGO: 출력기발, 1bit
IEN: 새치기가능비트, 1bit

전신타자기로부터의 건반입력과 전신타자기에로의 인쇄출력은 I/O 장치에 의하여 조종된다. 전신타자기는 8bit 단어로 영어자모와 수자기호들을 부호화하고 8bit 단어를 영어자모와 수자로 해신할수 있다.

ㄱ. 이 문제에서 서술된 첫 4 개의 등록기들을 리용하여 어떻게 처리장치가 전신타자기와 I/O 동작을 할수 있는가를 서술하시오.

ㄴ. 또한 IEN 을 리용함으로써 기능이 어떻게 더 효과적으로 수행될수 있는가를 서술하시오.

6. 그림 3-25 에서는 다중모선 I 에서 리용할수 있는 분산형중재방식을 보여 주었다. 모선요구자들은 우선순위에 따라 물리적인 련쇄사슬모양으로 접속되였다. 가장 왼쪽 요구자는 모선을 요구하는 더 우선권이 높은 요구자가 없다는것을 지적하는 신호(SPRN)에 의하여 제일 높은 모선우선권을 부여 받는다. 만일 요구자가 모선요구를 바라지 않는다면 자기의 모선우선권보내기신호를 설정한다. 박자주기의 시작에서 임의의 요구자는 자기의 모선우선권보내기(BPRN)신호를 능동으로 설정한다. 박자주기의 시작에서 임의의 요구자는 자기의 BPRO 를 낮은 준

위로 설정하여 모선조종권을 요구할수 있다. 이것은 자기의 BPRO 신호가 낮은 준위로 된것과 일치하도록 사슬내의 다음요구자의 BPRN 을 낮은 준위로 설정한다. 따라서 이 신호는 사슬을 따라 전파된다. 이 사슬작용의 끝에 BPRN 이 설정되고 BPRO 가 자기 사명을 못하는 하나의 모선요구자가 존재한다. 이 요구자는 우선권을 가진다. 만일 모선주기의 시작에서 모선이 동작하지 못하면 (비능동상태이면) 우선권을 가진 요구자는 BUSY 신호선을 설정함으로써 모선의 조종을 점유할수 있다.

BPR 신호를 가장 높은 우선권을 가진 요구자로부터 우선권이 가장 낮은 요구자까지 전송하기 위한 확실한 시간을 가진다. 이 시간은 주기박자보다 작아야 하는가? 실례를 들어 보시오.

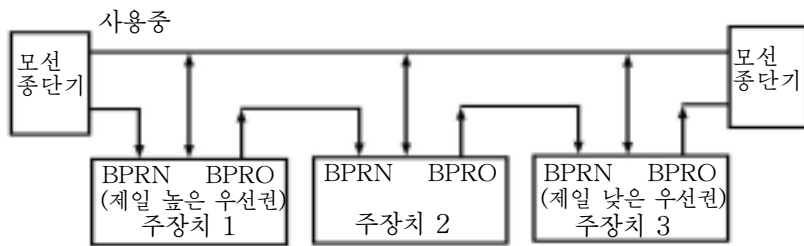



그림 3-25. 다중모선 ID 의 분산형중재원리

7. VAXSBI 모선은 분산형동기식중재방법을 리용한다. 매개 SBI 장치(실례로 처리장치, 기억기, 단일모선접속기)들은 하나의 우선권을 가지며 하나의 전송요구(TR)선을 가진다. SBI 는 이와 같은 16 개의 신호선(TR_0~TR_{15})을 가지며 여기서 TR_0 이 우선권이 제일 높다. 어떤 장치가 모선리용을 요구할 때 현재시간슬로트에서 자기의 TR 선들을 조사한다. 즉 예약된 가장 우선권이 높은 장치는 다음시간슬로트를 리용한다.

   최대로 17 개의 장치들이 모선에 접속될수 있다. 우선권이 16 인 장치는 TR 선이 없다. 왜 그런가?
8. 사실 가장 낮은 우선권을 가진 장치가 보통 가장 작은 평균기다림시간을 가진다. 이 리유로 처리장치는 보통 SBI 에서 가장 낮은 우선권으로 주어 진다. 왜 우선권이 16 인 장치는 가장 작은 평균기다림시간을 가지는가? 어떤 환경에서 사실이 맞지 않을수 있는가?
9. PCI 쓰기조작에 대한 시간선도를 그리고 설명하시오.

부록 3. 시간선도

이 장에서 시간선도는 사건들의 순서와 사건들사이의 의존관계를 설명하는데 리용되였다. 시간선도를 잘 모르는 독자들을 위하여 이 부록에서는 간단한 설명을 준다.

모선에 련결된 장치들사이의 통신은 신호들을 나르는 능력을 가진 선들의 묶음에 의하여 이루어 진다. 2 진수 0 과 1 로 표현되는 두개의 서로 다른 신호준위들이 전송된다.

시간선도는 시간의 함수로서 선에 신호준위를 보여 준다. 규약에 의하여 2진수 1신호준위는 2 진수 0 보다 높은 준위로서 묘사한다. 일반적으로 2 진수 0 은 암시적인 값이다. 즉 자료 혹은 다른 신호가 전송되지 않고 있다면 이때 선의 준위는 2진수 0을 표현하는 것이다. 0으로부터 1에로의 신호절환은 흔히 오름면으로서 표현한다면 1로부터 0으로의 절환은 내림면으로 표현된다. 명백히 신호절환은 흔히 급격히 발생하는것으로 표현된다. 실제로 절환시간은 령이 아닌 어떤 값을 가진다. 그러나 이 절환시간은 신호준위의 지속시간에 비하여 일반적으로 작다. 시간선도에서 지속되거나 불필요한 시간이 나타나는 사건들사이에 존재한다는것을 찾아 볼수 있다. 이것은 시간렬에서 공백으로 표현된다.

신호들은 때때로 그룹들로 표현된다. 실례로 자료가 한번에 한바이트 전송된다면 8개의 선이 요구된다. 일반적으로 이와 같은 그룹으로 전송되는 정확한 값을 아는것은 중요하지 않으며 오히려 신호들이 존재하는가 안하는가가 중요하다.

한개의 선에서 신호를 절환시키면 다른 신호선들에서 신호가 변하도록 덧붙어 있는 장치를 시동시킨다. 실례로 기억기가 읽기조종신호를 탐색하였다면 자료모선들에 자료신호들을 배치한다. 이와 같이 발생과 작용사이의 관계는 사건들의 순서를 만든다. 화살표는 이 의존관계를 보여 주기 위하여 시간선도에 리용된다(그림 3-26 ㄴ).

박자신호선은 흔히 체계모선에 포함된다. 전자적박자는 박자신호선에 련결되며 반복적이며 규칙적인 순서렬을 제공한다. 다른 사건들은 박자신호에 동기되여야 한다.

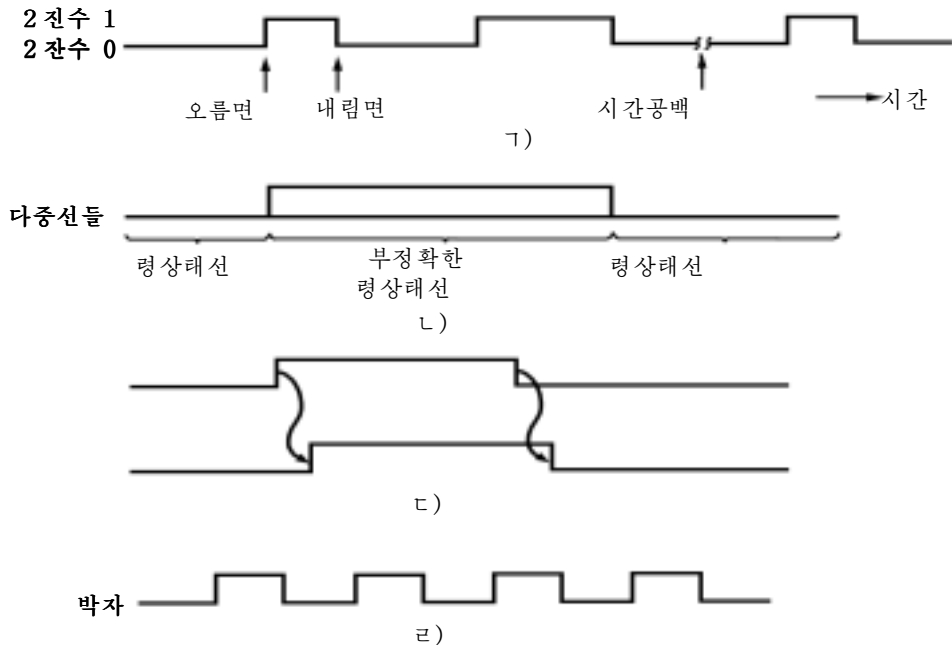
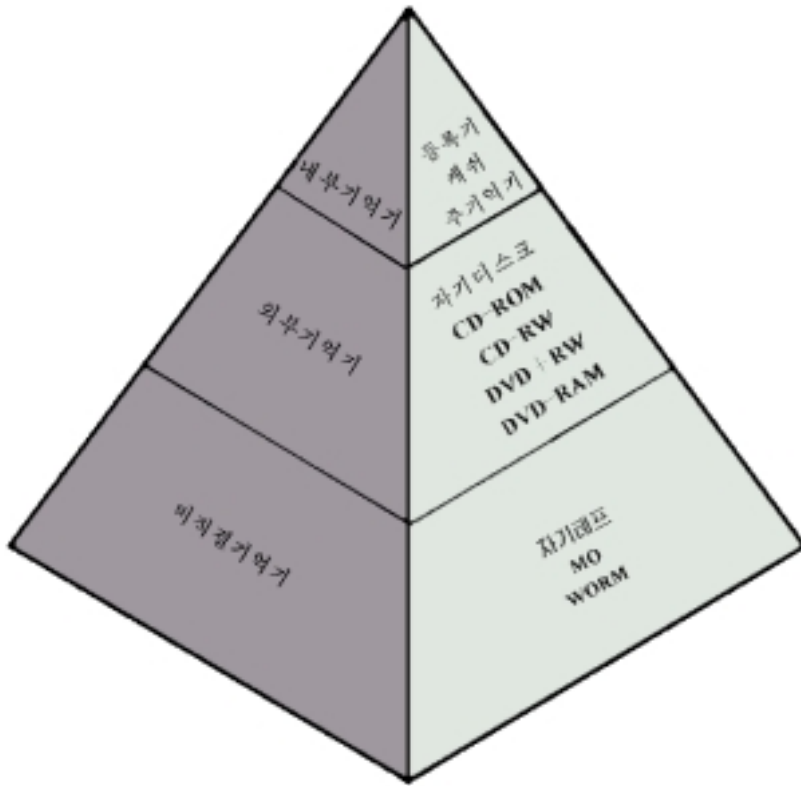



그림 3-26. 시간선도

ㄱ-시간함수로서의 신호, ㄴ-다중선, ㄷ-원인과 결과의 의존관계, ㄹ-박자신호

제 4 장. 내부기억기

◆◆

- 콤퓨터기억기는 계층으로 이루어 진다. 가장 높은 준위는 처리장치등록기들이다. 다음준위는 단일캐쉬와 L1 과 L2 로 표시되는 2 준위캐쉬이다. 그다음준위는 일반적으로 DRAM 으로 구성되는 주기억기이다. 이 모든것들은 콤퓨터체계에서 내부기억기로 고찰된다. 기억기의 계층은 외부기억기로 확장되며 다음준위는 일반적으로 하드디스크이고 그다음은 ZIP 디스크, 빛디스크와 자기테프 등 여러개의 준위로 구성된다.
- 이렇게 기억기의 계층을 구성하면 비트당 가격을 줄이고 용량을 크게 하며 호출시간을 줄인다. 하나의 최고속기억기로만 리용하는것이 좋지만 값이 비싸기때문에 속도가 뜬 기억기들을 많이 리용하여 허용한도와 호출시간을 유지하면서 가격을 눅게 한다. 이 기교는 요구되는 기억기단어를 일반적으로 더 빠른 기억기에서 호출하도록 기억기에 자료와 프로그람을 구성하는것이다.
- 일반적으로 처리장치가 주기억기를 앞으로 맨 먼저 호출할 주소는 맨 마지막으로 호출했던 주소라고 보는것이 좋다. 그러므로 캐쉬는 DRAM 으로부터 가장 최근에 리용된 일부 단어들에 대한 복사를 자동적으로 진행한다. 만일 캐쉬가 적당히 설계되였다면 이때에 대부분의 처리장치는 이미전에 캐쉬에 있는 기억기단어들을 요구한다.

◆◆

얼핏 보기에는 간단한것 같지만 콤퓨터기억기는 대체로 콤퓨터체계의 몇가지 특성들인 형, 기술, 구성, 성능, 가격들과 같은 넓은 범위를 고려하여 실현된다. 콤퓨터체계의 기억기요구를 만족시키는 최량의 기술은 없다. 그러므로 전형적인 콤퓨터체계는 일부 내부적인 기억기보조체계(처리장치에 의하여 직접 호출가능한)와 일부 외부적인 기억기보조체계(I/O 장치를 통하여 처리장치가 호출가능한)들로 계층을 이룬다.

이 장은 내부기억구성요소들에 중점을 두며 한편 제 5 장은 외부기억체계를 서술한다. 제 1 절에서는 콤퓨터기억기의 기본특성체계를 서술한다. 그다음 주기억으로 리용되는 ROM, DRAM, SRAM 과 같은 반도체기억기들을 고찰한다. 다음으로 모든 현대적콤퓨터체계들의 본질적인 구성요소인 캐쉬기억기를 설명한다. 마지막으로 현대적인 DRAM 방식을 고찰한다.

제 1 절. 콤퓨터기억기체계의 일반개념

1. 기억기체계의 특성

만일 기억기들의 기본특성들에 따라 기억체계들을 갈라 놓으면 콤퓨터기억기의 복합체는 더 관리하기가 쉽다. 이것들의 가장 중요한 점들을 표 4-1 에 보여 주었다.

표 4-1 에서 **위치**라는 말은 기억기가 콤퓨터의 내부에 있는가 외부에 있는가 하는 것을 의미한다. 내부기억기는 흔히 주기억기라고 한다. 그러나 내부기억기에는 다른 형태들도 있다. 처리장치는 등록기형태로 자기의 국부기억기를 요구한다(실례로 그림 2-3 참고). 더우기 처리장치의 조종장치부분들도 자기의 내부기억기를 요구한다. 다음장들에서 이 두가지 형태들에 대하여 론의한다. 캐쉬는 내부기억기의 또 하나의 다른 형태이다. 외부기억기는 처리장치가 I/O 조종기들을 통하여 호출할수 있는 디스크나 자기테프와 같은 주변기억장치들로 구성된다.

표 4-1. 콤퓨터기억기체계의 기본특성

위치	**성능**
처리장치	호출시간
내부(1 차)	주기시간
외부(2 차)	전송속도
용량	**물리적인 형태**
단어크기	반도체
단어수	자기
전송단위	빛
단어	자기-빛
블로크	**물리적특성**
호출방법	휘발성/비휘발성
순차	지우기가능/지우기불가능
직접	**구성**
자유	
련상	

기억기의 명백한 특성은 **용량**이다. 내부기억기에 대하여 그 특성은 일반적으로 바이트(1byte=8bit) 혹은 단어라는 용어로 표현된다. 일반적인 단어길이는 8, 16, 32bit 이다. 외부기억기용량은 일반적으로 바이트에 의하여 표현된다.

련관된 개념의 하나는 **전송단위**이다. 내부기억기에 대하여 전송단위는 기억기로부터 혹은 기억기모듈에로의 자료선의 수와 같다. 이것은 단어길이와 흔히 같지만 그렇지 않은 경우도 있다. 이 점을 명백히 하기 위하여 내부기억기에 대한 3 가지 련관개념들을 고찰하자.

- **단어:** 기억기조직의 기본단위이다. 단어의 크기는 일반적으로 수를 표시하고 명령길이에 리용되는 비트의 수와 같다. 그러나 그렇지 못한 경우도 있다. 실례로 CRAY-1은 64bit 단어길이를 가지지만 24bit 옹근수표현형식을 리용한다. VAX는 여러개의 바이트들로 이루어 지는 다양한 길이의 명령들을 가지며 32bit 의 단어크기를 가진다.
- **주소화단위:** 많은 체계들에서 주소화단위는 단어이다. 그러나 일부 체계들은 바이트준위에서 주소화를 허가한다. 주소의 비트길이가 A이고 주소화단위의 수가 N인 경우 그 사이의 관계는 2^A=N이다.
- **전송단위:** 주기억기에서 이것은 한번에 주기억기에 대한 읽기 혹은 쓰기를 위한 비트수이다. 전송단위의 요구는 단어 혹은 주소화단위와 같지 않다. 외부기억기에서 자료는 흔히 단어보다 더 긴 단위로 전송되며 이것을 블로크라고 한다.

기억기형식을 구별하는 또 하나의 방법은 자료의 단위들을 호출하는 방법이다. 여기에는 다음과 같은것들이 포함된다.

- **순차호출:** 기억기는 레코드라고 하는 자료의 단위들로 구성된다. 호출은 지정된 선형순서로 진행되여야 한다. 기억된 주소화정보들은 개별적인 레코드들에 대하여 리용되며 회복과정도 같다. 공유된 읽기/쓰기기구를 리용하며 이것은 현재위치로부터 목적위치까지 중간레코드들을 무시하고 통과시키면서 이동시켜야 한다. 따라서 임의의 레코드들을 호출하는 시간은 매우 다양하다. 테프장치들은 순차호출로 진행된다.
- **직접호출:** 순차호출처럼 직접호출은 공유된 읽기-쓰기기구를 포함한다. 그러나 개별적인 블로크 혹은 레코드들은 물리적위치에 기초한 단일한 주소를 가진다. 주어 진 부분에 대하여 직접호출로서 완성되는 기억기호출은 최종위치에 도달할 때까지 탐색, 계수, 기다림의 순서로 이루어 진다. 그러므로 호출시간은 서로 다르다. 디스크장치는 직접호출방식이다.
- **자유호출:** 기억기에서 모든 주소화가능한 위치는 유일하게 물리적배선에 의한 주소화기구를 가진다. 주어 진 위치를 호출하는 시간은 선행한 호출들의 순서에 무관계하며 상수이다. 따라서 임의의 위치는 자유적으로 선택할수 있으며 직접적으로 주소화하고 호출할수 있다.주기억기와 일부 캐쉬체계는 자유호출기억기다.
- **련상호출:** 이것은 지적된 단어배렬중의 한 단어안에서 요구되는 비트의 비교를 진행하며 모든 단어들에 대하여 동시에 이 조작을 진행하여 어느 하나를 선택하는 자유호출형태의 기억기이다. 따라서 단어는 주소에 의해서가 아니라 내용의 일부분에 의하여 호출된다. 보통 자유호출기억기에서처럼 모든 위치는 자기의 주소화기구를 가지며 호출시간은 위치 혹은 선행패턴호출들과 독립적으로 일정하다. 제3절에서 론의하는 캐쉬기억기는 련상호출을 채용한다.

사용자견해로부터 기억기의 가장 중요한 두가지 특성은 용량과 성능이다. 성능에는 다음의 세가지 파라메터들이 리용된다.

- **호출시간:** 자료호출기억기에 대하여 호출시간은 읽기 및 쓰기조작을 수행하는데 걸리는 시간이다. 즉 주소가 기억기에 주어 진 순간부터 자료가 기억되거나 리용가능하게 될 때까지의 시간이다. 비자유호출기억기에 대하여 호출시간은 요구하는 위치에서 읽기-쓰기기구를 배치하는데 걸리는 시간이다.
- **기억기주기시간:** 이 개념은 기본적으로 자유호출기억기에 대하여 적용되며 두번째 호출이 개시되기전에 요구되는 추가적인 시간에 호출시간을 더한것으로 이루

어 진다. 이 추가적인 시간은 자료를 파괴적으로 읽었을 때 신호선들을 안정시키거나 자료들을 재생하는 시간이 요구된다.

- **전송속도:** 이것은 자료가 기억장치에로 혹은 기억장치로부터 전송되는 속도이다. 자유호출기억기에 대하여 이것을 1/주기시간과 같다.

비자유호출기억기에 대하여 아래와 같은 관계가 성립한다.

$$T_N = T_A + \frac{N}{R}$$

여기서 T_N – N 개의 비트를 읽기 혹은 쓰기 위한 평균시간,
T_A – 평균호출시간,
N – 비트수,
R – 전송속도(bps)

기억기의 다양한 **물리적형태**가 사용되고 있다. 오늘 가장 일반적인것은 반도체기억기, 디스크와 자기테프에 리용되는 자성면기억기, 빛과 자기빛기억기들이다.

자료기억의 여러가지 **물리적특성**들은 중요하다. 휘발성기억기에서 전원이 차단되였을 때 정보는 본질적으로 잃게 된다. 비휘발성기억기에서 일단 기억된 정보는 고의적으로 변경시키지 않는 한 유지된다. 즉 정보를 유지하는데 전원을 요구하지 않는다. 자성면기억기는 비휘발성기억기이다. 지울수 없는 기억기는 기억장치파괴를 제외하고 변경될수 없다. 이러한 형태의 반도체기억기를 ROM(read only memory)이라고 한다. 실제적으로 지울수 없는 기억기는 비휘발성기억기이다.

자유호출기억기에 대하여 **구성**은 기본설계지표이다. 구성에 의하여 단어를 형성하는 비트들의 물리적배렬을 결정한다. 얼마 후에 설명하는바와 같이 명백한 배렬이 항상 리용되지 못한다.

2. 기억기의 계층

콤퓨터의 기억기를 제한하는 설계는 세가지 질문 즉 얼마나 용량이 커야 하는가, 얼마나 빨라야 하는가, 얼마나 비용이 많이 드는가에 의하여 종합된다.

얼마나 용량이 커야 하는가하는 질문은 비교적 해결되였다. 용량에 대하여 말한다면 응용프로그람들은 용량의 크기에 맞게 개발될것이다. 속도가 얼마나 빠른가하는 질문은 어떤 의미에서 쉬운 대답이다. 더 높은 성능을 달성하기 위하여 기억기는 처리장치속도에 뒤떨어 지지 말아야 한다. 즉 처리장치가 명령들을 집행하고 있을 때 명령들과 연산수들에 대한 기다림으로 하여 정지되지 말아야 한다. 마지막질문도 고찰하여야 한다. 실제로 체계에서 기억기의 가격은 다른 구성요소들에 비하여 알맞춤하여야 한다.

상품화에서는 기억기의 세가지 기본특성 즉 가격, 용량, 호출시간 등이 알맞을것을 요구한다. 주어 진 임의의 시점에서는 여러가지 기술들이 기억기체계들을 실현하는데 리용된다. 기술들의 이 부분들은 서로 교차되며 다음관계가 성립한다.

- 호출시간을 빠르게 하면 비트당 가격이 높아 진다.
- 용량을 크게 하면 비트당 가격이 눅어 진다.
- 용량을 크게 하면 호출시간은 떠진다.

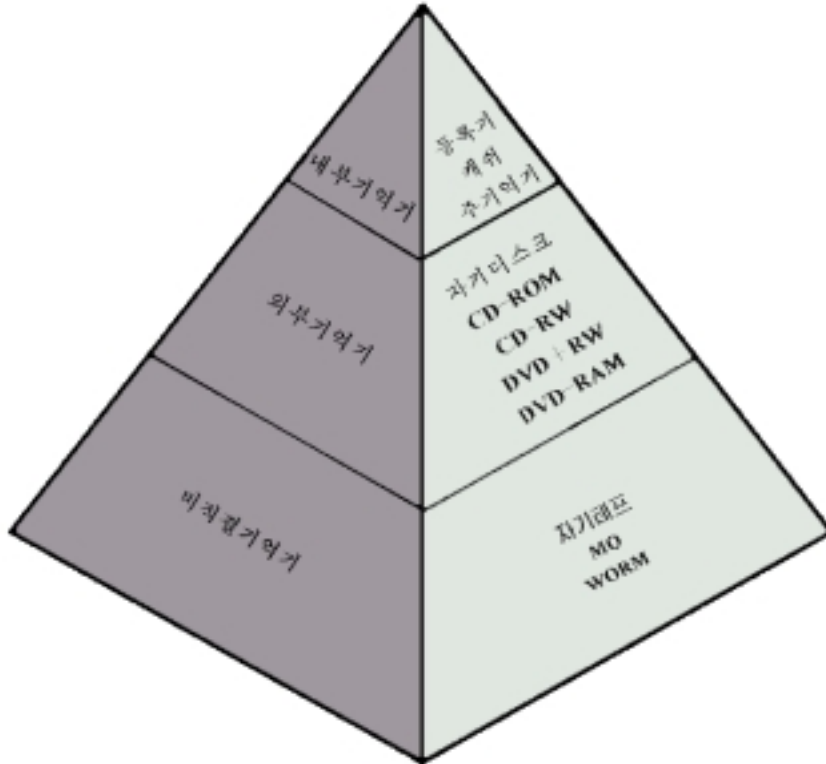



그림 4-1. 기억기의 계층

설계자의 고충은 명백하다. 설계자는 용량을 만족시키면서 비트당 가격이 눅은 대용량기억기를 제공하는 기억기기술들을 리용할것을 바란다. 그러나 성능요구에 대하여 설계자는 비싼 비용을 들여 빠른 호출시간을 달성할수 있는 상대적으로 적은 용량의 기억기를 설계하게 된다.

이 고충을 해결하는 방법은 단일기억기구성 혹은 기술로 해결하지 못한다. 그러나 **기억기의 계층화**를 실현하면 해결할수 있다. 표준적인 계층화를 그림 4-1 에 주었다. 계층화를 실현할 때 다음과 같은 문제들이 발생한다.

ㄱ. 비트당 가격이 감소한다.
ㄴ. 용량이 증가한다.
ㄷ. 호출시간이 증가한다.
ㄹ. 처리장치에 의한 기억기의 호출주기는 감소된다.

따라서 용량이 작고 많은 비용이 들며 빠른 기억기는 용량이 크고 비용이 적게 들고 속도가 뜬 기억기에 의하여 보상된다. 이 구성의 성공적인 열쇠는 ㄹ조건이다. 즉 호출시간의 감소이다. 이 장의 뒤에서 캐쉬기억기와 제 7 장에서 가상기억기를 론의할 때 더 상세한 개념을 설명한다. 여기에서는 간단히 설명한다.

처리장치가 2개 준위의 기억기에 대한 호출을 진행한다고 가정하자. 1준위는 1000개의 단어를 가지고 있고 0.1 μs 의 호출시간을 가지며 2준위는 100000개의 단어를 가지고 있고 1 μs 의 호출시간을 가진다. 만일 호출되는 단어가 1 준위에 있다고 가정하면 그때 처리장치는 그것을 직접 호출한다. 만일 단어가 2 준위에 있으면 단어는 처음

에 1 준위에로 전송되며 그다음 처리장치에 의하여 호출된다. 간단히 하기 위하여 처리장치가 단어가 1 준위에 있는가 2 준위에 있는가를 검사하는데 요구되는 시간을 무시하자. 그림 4-2 에서는 이 조건을 무시한 그라프의 형태를 보여 주었다. 이 그림은 명중률 H의 함수로서 두 준위기억기에 대한 평균호출시간을 보여 준다. 여기서 H는 고속기억기(실례로 캐쉬기억기)에서 명중된 모든 기억기호출비률의 역수로서 정의되며 T_1 는 1준위에서 호출시간이며 T_2은 2준위에서의 호출시간이다. 보는바와 같이 1준위 호출의 높은 비률에 대하여 평균 총 호출시간은 2준위의것보다 1준위의것에 더 가깝다.

실례에서 기억기호출의 95%가 캐쉬에서 명중되였다고 가정하자. 이때 한 단어의 평균호출시간은 다음과 같이 계산된다.

$$(0.95)(0.1\ \mu s)+(0.05)(0.1\ \mu s+1\ \mu s)=0.095+0.055=0.15\ \mu s$$

조건 ㄱ로부터 ㄹ까지가 적용된다면 이러한 원리로 작업한다. 여러가지 기술을 채용한것에 의한 기억기체계들의 변화는 조건 ㄱ로부터 ㄷ까지 만족하게 하였다. 다행히도 조건 ㄹ는 일반적으로 타당하다.

조건 ㄹ의 타당성에 대한 기본근거는 참조의 **국부성**이라고 하는 원리이다 [DENN68]. 프로그람집행과정에 기억기는 처리장치에 의하여 참조되며 명령들이나 자료들은 기억기에 밀집되여 있다. 프로그람들은 일반적으로 여러번 반복되는 순환과 보조프로그람들을 포함하며 일단 순환고리나 보조프로그람이 집행되면 작은 명령모임들에 대한 참조를 반복한다. 마찬가지로 표들과 배렬들에 대한 연산들은 밀집된 자료단어들의 묶음에 대한 호출을 진행한다. 오랜 시간 지나면 리용되는 클라스터(밀집된 부분)들은 변하지만 짧은 시간동안에는 처리장치가 기억기의 고정된 클라스터들에서 작업한다.

그러므로 모든 련속적인 낮은 준위에 대한 호출비률이 웃준위에 대한 호출비률보다 실제적으로 작으므로 계층에 따라 자료를 구성하는것이 가능하다. 이미 고찰한 2개 준위의 실례를 보자. 2준위기억기는 모든 프로그람명령들과 자료들을 다 포함한다. 현재 클라스터들은 1 준위에 림시적으로 배치되여 있다. 시간이 지나면 1 준위로 들어오는 새로운 클라스터를 배치할수 있게 1 준위에 있는 클라스터들중 어느 한개는 2 준위로 다시 복귀된다. 그러나 평균적으로 대부분 참조는 1 준위에 있는 명령과 자료들이다.

이 원리는 그림 4-1 에서 보여 준 계층에서 알수 있는바와 같이 2개이상의 준위를 가진 기억기체계들에 적용할수 있다. 더 빠르고 더 용량이 작으며 더 비용이 많이 드는 기억기형태는 처리장치내부에 있는 등록기들이다. 일반적으로 처리장치는 한타스정도의 등록기를 가지며 일부 처리장치들인 경우에는 수백개의 등록기들을 가진다. 간단히 2 개 준위로 하면 주기억기는 콤퓨터의 기본내부기억기이다. 주기억기에서 모든 기억위치는 유일한 주소이며 대부분의 기계명령들은 하나이상의 주기억주소들에 포함된다. 주기억기는 보통 더 속도가 빠르고 용량이 작은 캐쉬로 확장한다. 캐쉬는 프로그람작성자에 의하여 실제로는 처리장치에 의하여 볼수 없다. 성능을 개선하기 위하여 주기억기와 처리장치등록기들사이에 자료전송을 보장하기 위한 장치이다.

지금까지 서술한 기억기의 세가지 형태는 휘발성이며 반도체기술을 도입하였다. 이 3 개준위들의 리용은 반도체기억기가 가격과 속도가 서로 다른 여러가지 형태들로 되여 있다는 점을 고려한다. 자료는 하드디스크나 비직결형매체들 즉 디스크, 자기테프, 빛기억기와 같은 대용량외부기억장치들에 영구적으로 기억된다. 외부적인 비휘발

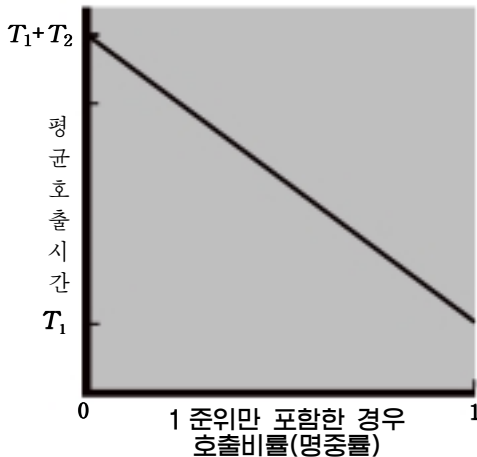
성기억기를 2 차기억기 혹은 보조기억기라고도 한다. 이것들은 프로그람과 자료파일들을 기억하는데 리용되며 보통 개별적인 바이트와 단어들에 대해서가 아니라 파일이나 레코드라는 용어로만 프로그람작성자에 의하여 볼수 있다. 디스크는 또한 제 7 장에서 론의하는 가상기억기로서 주기억기확장을 제공한다.

기억기의 다른 형태들은 기억기계층에 포함된다. 실례로 대형 IBM 콤퓨터들은 확장기억기로 알려 진 내부기억기를 리용한다. 이 기억기는 주기억기보다 속도가 뜨고 비용이 적게 드는 반도체기술을 리용한다. 엄격히 말하면 이 기억기는 기억기계층에 적합치 않지만 한 측면부분을 이룬다. 자료는 주기억기와 확장기억기사이에 전송될수 있지만 확장기억기와 외부기억기사이에는 전송될수 없다. 2 차기억기의 또 다른 형태들은 빛디스크와 자기빛디스크이다. 마지막으로 더 추가되는 준위들은 프로그람에서 기억기계층에 효과적으로 첨부될수 있다. 주기억기의 일부는 디스크로부터 읽어 진 자료를 림시적으로 보관하는 완충기로서 리용될수 있다. 이와 같은 기술은 때때로 디스크캐쉬[1]이라고 하며 다음의 두가지 방법으로 성능을 개선한다.

그림 4-2. 간단한 2 준위 기억기의 성능

- 디스크쓰기들을 묶어서 진행한다. 이렇게 하면 수많은 자료의 작은전송들을 어느 정도 큰 자료전송으로 만든다. 이것은 디스크의 성능을 개선하며 디스크에 대한 처리장치의 관련성을 최소로 한다.
- 출력으로 미리 정해 진 일부 자료들은 디스크에 기억되기전에 프로그람에 의하여 참조될수 있다. 이 경우에 디스크로부터 느리게 참조되기보다는 프로그람 캐쉬로부터 빨리 참조될수 있다.

부록 4-1 에서는 여러 준위기억기구조의 밀접한 성능관계들을 설명하였다.

제 2 절. 반도체주기억기

종래의 콤퓨터들에서 콤퓨터주기억기로서 리용한 대부분의 자유호출기억기의 일반형태는 **자심**기억기라고 하는 원통모양의 자기철심고리들의 배렬이였다. 그리하여 주기억기를 흔히 **자심**기억기라고 하였다. 미소전자공학의 출현과 실현은 자심기억기를 성공한 때로부터 긴 세월이 걸렸다. 오늘 주기억기를 위한 반도체소편들의 리용은 거의 공통이다. 이 기술의 기본내용들을 이 절에서 설명한다.

---

[1] 디스크캐쉬는 일반적으로 하나의 순수한 쏘프트웨어 기술인데 이책에서는 론의하지 않겠다. [STAL98]을 참고

1. 자유호출반도체기억기의 분류

이 절에서 제공하는 모든 기억기형태들은 자유호출기억기이다. 즉 기억기의 개별적인 단어들은 배선된 주소론리를 통하여 직접 호출된다.

표 4-2 는 반도체기억기의 기본형태들을 주었다. 대부분의 보편적인것들을 RAM (random-access memory) 이라고 한다. 물론 이것은 표에서 모든 형태들이 자유호출이므로 잘못 쓰인 용어이다. RAM 의 구별되는 특성은 기억기로부터의 자료읽기와 기억기에로의 자료쓰기를 쉽고 빠르게 할수 있다는것이다. 읽기와 쓰기는 전기적신호에 의하여 달성된다.

표 4-2. 반도체기억기의 형태

기억기의 형태	부류	지우기	쓰기방식	휘발성
자유호출기억기(RAM)	읽기-쓰기기억기	전기적으로 바이트준위	전기적으로	휘발
읽기전용기억기(ROM)	읽기전용기억기	불가능	(금지됨)	비휘발
프로그람가능한 ROM(PROM)			전기적으로	
소거가능한 PROM(EPROM)	읽기기본기억기	자외선, 소편준위		
플라쉬기억기		전기적으로, 블로크준위		
전기적으로 소거가능한 PROM(EEPROM)		전기적으로, 바이트준위		

RAM 의 다른 구별특성은 휘발성이다. RAM 은 정상적으로 전원을 공급하여야 한다. 전원이 중단되면 자료는 잃어 버린다. 따라서 RAM 은 오직 림시적인 기억기로서만 리용될수 있다.

RAM 기술은 두개의 기술. 즉 정적기억기, 동적기억기로 나누어 진다. **동적 RAM** 은 축전기의 충전과정으로서 자료를 기억하는 세포로 만들어 진다. 축전기에서 충전 혹은 방전은 2 진수 1 혹은 0 으로서 표현된다. 축전기들은 저절로 방전되는 경향성을 가지며 동적 RAM 들은 자료기억을 유지하기 위한 재생동작인 주기적인 충전을 요구한다. 정적 RAM 에서 2 진값들은 전통적인 방아쇠론리문구성들을 리용하여 기억한다(방아쇠의 서술에 대하여 부록 I 참고). 정적 RAM 은 전원이 공급되는 동안에만 자료를 보존한다.

정적 RAM 과 동적 RAM 들은 휘발성이다. 동적기억기세포들은 간단하며 정적기억기세포보다 작다. 따라서 동적RAM은 밀도가 높으며 정적RAM에 비하여 비용이 적게 든다. 다른 한편 동적 RAM 은 재생회로를 지원할것을 요구한다. 큰 기억기의 경우에 재생회로의 가격이 고정되여 있으므로 동적기억기세포들의 작은 가격변화에 의하여 보상되는것보다 더 리득이다. 따라서 동적기억기는 큰 기억기가 필요한 곳에 쓰는것이 좋다. 마지막문제는 정적 RAM 들이 일반적으로 동적 RAM 들보다 좀 더 빠르다는것이다.

RAM 에 대조되는것은 ROM(read only memory)이다. 이름이 말해 주는것처럼 ROM 은 변화시킬수 없는 영구적인 자료패턴들을 포함한다. ROM 을 기본적으로 응용

하는것은 제 4 편에서 론의하는 마이크로프로그람조종방식이다. 다른 가능한 적용은 다음과 같다.

- 자주 쓰이는 기능들에 대한 서고보조루틴
- 체계프로그람
- 함수표

적당한 크기의 요구에 대하여 ROM 의 우점은 자료 혹은 프로그람이 항시적으로 주기억기에 있으며 2 차기억기로부터의 넣기를 요구하지 않는다는것이다.

ROM 은 다른 집적회로소편들과 마찬가지로 제조공정의 한 부분으로서 자료를 실제적으로 소편속에 삽입하여 만든다. 이것은 다음의 두가지 문제를 제기한다.

- 자료삽입단계는 비교적 높은 고정된 가격을 포함하며 개별적인 ROM 에 대하여 하나 혹은 수천개의 복사물이 제작된다.
- 오유가 발생할 공간이 없다. 만일 한 비트가 오유라면 전체 ROM 묶음은 버려야 한다.

개별적기억내용을 가진 적은 량의 ROM 이 요구될 때 비용을 적게 들일수 있는 한가지 방법은 프로그람가능한 ROM(PROM)을 리용하는것이다. ROM 과 같이 PROM 은 비휘발성기억기이며 한번 쓰기를 진행할수 있다. PROM 에 대하여 쓰기과정은 전기적으로 진행하며 기본소편제작후에 공급자나 거래자에 의하여 수행된다. 쓰기 혹은 프로그람화과정을 위하여 특수한 기구가 요구된다. PROM 들은 유연성과 편리성을 제공한다. ROM 은 높은 생산성으로 하여 인기를 가지고 있다.

ROM 에서 또 하나의 변종은 읽기조작이 쓰기조작보다 더 자주 진행되지만 비휘발성기억기를 요구하는 응용프로그람에 대하여 효과적인 read-mostly memory(읽기기본기억기)이다. 여기에는 읽기기본기억기로서 3 가지 일반적형식들 즉 EPROM, EEPROM, flash(고속지우기 ROM)기억기가 있다.

빛소거형프로그람가능한 읽기전용기억기(EPROM)는 PROM 과 같이 전기적인 읽기와 쓰기를 진행한다. 그러나 쓰기조작전에 모든 기억세포는 자외선으로 소편을 비쳐서 초기상태와 같이 지우기를 해야 한다. 이 지우기과정은 반복하여 진행할수 있다. 즉 매번 지우기를 끝내려면 길어서 20 분정도 걸릴수 있다. 따라서 EPROM 은 여러번 변경시킬수 있으며 ROM 이나 PROM 과 같이 실제적으로 명백하게 자료를 유지하지 못한다. 기억의 원가를 비교하면 EEPROM 은 PROM 보다 더 많은 비용이 들지만 여러번 갱신할수 있는 능력을 가진다는 우점이 있다.

읽기기본기억기의 더 능률적인 형태는 **전기소거형프로그람가능한 읽기전용기억기**(EEPROM)이다. 이 기억기는 이미 존재하던 내용을 지우지 않고 임의의 순간에 쓰기할수 있는 읽기기본기억기이다. 즉 주소화된 한바이트 혹은 여러개의 바이트들에 대하여서만 갱신될수 있다. 쓰기조작은 읽기조작보다 상대적으로 더 길다. 즉 바이트당 수백 μs 걸린다. EEPROM 은 정상적으로 설치되여 있는 모선조종, 주소와 자료모선을 리용하여 콤퓨터에 설치된 상에서 갱신할수 있는 유연성과 비휘발성의 우점들을 가지고 있다. EEPROM 은 EPROM 보다도 비용이 많이 들며 밀도가 높지 못하다. 즉 소편당 비트수가 작다.

반도체기억기의 또 하나의 다른 형태는 **플라쉬기억기**(프로그람갱신속도가 빠르다는데로부터 유래된 말이다.)이다. 1980 년대 중엽에 처음으로 소개되였으며 플라쉬기억기는 가격과 기능적으로 보면 EPROM 과 EEPROM 사이에 중간 정도로 된다.

EEPROM 과 같이 플라쉬기억기는 전기소거기술을 리용한다. 플라쉬기억기의 전체 내용은 1 ～수초사이에 지울수 있으며 따라서 EPROM 보다 고속이다. 또한 소편전체가 아니라 기억기의 블로크단위로 지우는것이 가능하다. 그러나 플라쉬기억기는 바이트준위의 지우기를 진행할수 없다. EPROM 과 같이 플라쉬기억기는 오직 비트당 한개의 3 극소자를 리용하며 이로부터 EPROM 과 같이 높은 밀도를 달성할수 있다

2. 조직

반도체기억기의 기본요소는 기억기세포이다. 비록 다양한 전자기술들이 리용되지만 모든 반도체기억기세포들은 다음의 속성들을 공동으로 가진다.

- 2 진수 1 과 0 을 표현하는데 리용할수 있는 두개의 안정한 상태를 가진다.
- 상태를 설정하기 위한 쓰기능력을 가진다.
- 상태를 읽기 위한 능력을 가진다.

그림 4-3에는 기억기세포의 조작을 제시하였다. 가장 일반적으로 세포는 전기적신호를 전송하는 능력을 가진 3 개의 기능적인 단자들을 가진다. 이름이 암시하는바와 같이 선택단자는 읽기 혹은 쓰기조작을 위한 기억기세포를 선택한다. 조종단자는 읽기인가 쓰기인가를 지적한다. 쓰기에 대하여 나머지단자는 1 혹은 0 으로 세포의 상태를 설정하는 전기적신호를 제공한다. 읽기에 대하여 이 단자는 세포의 상태를 출력하는데 리용된다. 기억기세포의 내부적구성과 기능, 동기화의 세부고찰은 리용되는 지적된 집적회로기술에 의존되며 이 책의 범위를 초과하는것으로 된다. 목적상 주어 진 개별적인 세포들을 읽기와 쓰기조작을 위하여 선택할수 있다는것을 알면 된다.

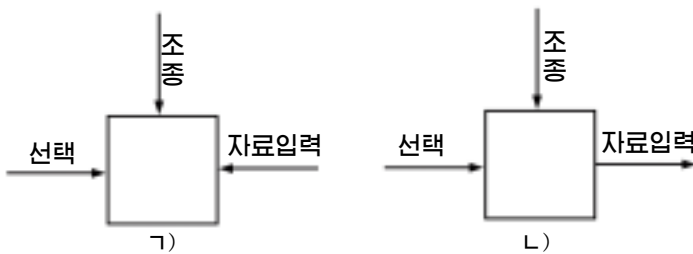



그림 4-3. 기억기세포조작
ㄱ-쓰기 ㄴ-읽기

3. 소편론리

다른 집적회로소편들과 마찬가지로 반도체기억기는 소자들로 제작된다. 매 소자들은 기억기세포들의 배렬들로 이루어 진다.

전체 기억기계층에서 속도, 용량, 가격과 같은 상품지표들이 있다는것을 보았다. 이 상품지표들은 한 소편에서 기억기세포들의 구성과 론리적기능들을 고찰할 때도 필요하다. 반도체기억기들에 대하여 기본설계지표의 하나는 동시에 읽기/쓰기할수 있는 자료의 비트수이다. 하나의 극단적인 방법은 기억기에서 단어들의 론리적인 배렬과 같게 물리적세포들의 배렬을 구성하는것이다. 배렬은 B 개의 비트들로 이루어 진 W 개의 단어들이 구성된다. 실례로 16Mbit 소편은 1M 개의 16bit 단어들로 구성될수 있다. 다른 극단적인 방법은 자료가 한번에 1bit 씩 읽기/쓰기되는 1bit 소편구성이라는것이다. DRAM 과 같은 기억기소편구성을 말할수 있다. ROM 구성도 비슷하다.

그림 4-4 에서는 16Mbit DRAM 의 일반적인 구성을 보여 주었다. 이 경우에 4bit 가 동시에 읽기 혹은 쓰기된다. 론리적으로 기억기배렬은 2048 × 2048 요소들의 4 각형

배렬로 구성된다. 여러가지 물리적배렬들도 가능하다. 임의의 경우에 배렬의 요소들은 수평(행)과 수직(렬)선들에 의하여 련결된다.모든 수평선들은 자기 행에 있는 모든 세포들의 선택단자에 련결된다. 모든 수직선들은 자기렬에 있는 모든 소편들의 Data-In/Sense 단자들에 련결된다.

주소선들은 선택되는 단어의 주소를 제공한다. 총 $\log_2 W$ 의 선들이 요구된다. 실례로 11개의 주소선들은 2048개행들의 하나를 선택할수 있게 한다. 이 11개의 선들은 11 개의 입구와 2048 개의 출구를 가진 행해신기에 입구된다. 해신기의 론리는 11 개의 입구선들에서 주어 지는 비트렬에 따라 2048 개의 출구가운데서 한개를 능동상태로 만

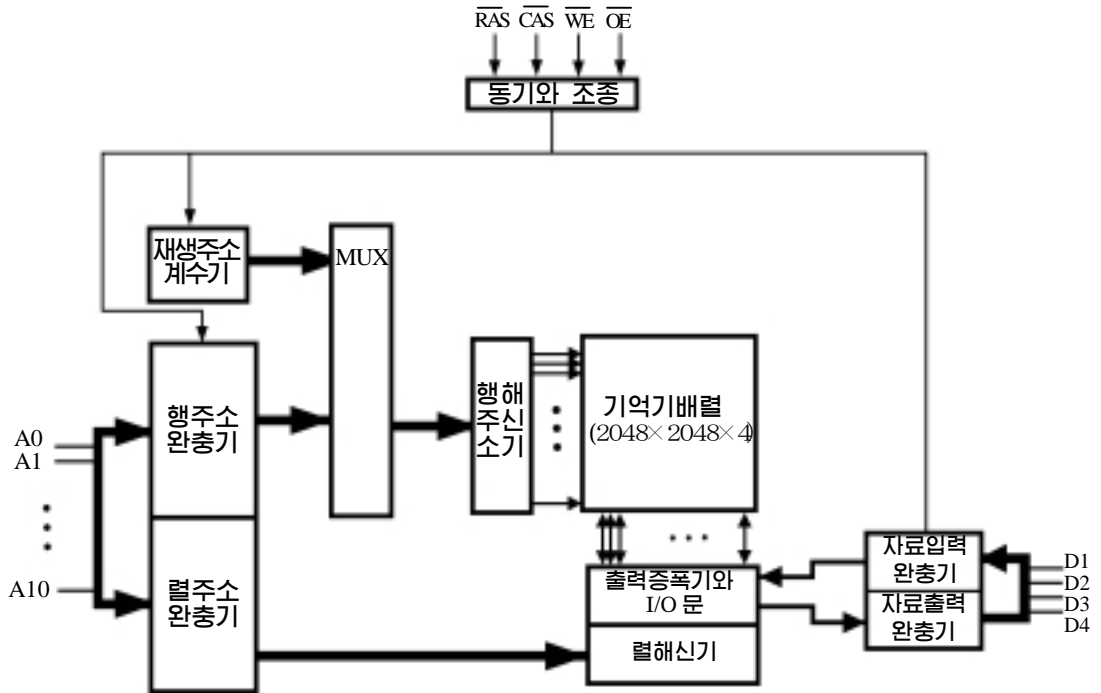



그림 4-4. 일반적인 16Mbit DRAM(4M×4)

든다(2^{11}=2048).

추가적으로 입력되는 11 개의 주소선들은 렬마다 4bit 로 구성된 2048 개의 렬들가운데서 한개를 선택한다. 4 개의 자료선들은 자료완충기와 4bit 입구와 출구로 리용된다. 입력(쓰기)에서 모든 비트선들에 대한 구동기는 자료선에 대응하는 값들에 일치하는 1 혹은 0 으로 동작한다. 출력(읽기)에서 모든 비트선들의 값은 읽기증폭기를 통과하여 자료모선으로 나간다. 행선은 행의 세포들이 읽기 혹은 쓰기에 리용되도록 선택한다.

이 DRAM 에서 4 개의 비트들로만 읽기/쓰기되므로 모선에서 한 자료단어를 읽기/쓰기 할수 있게 기억기 조종기에 여러개의 DRAM 들을 련결하여야 한다.

2048 × 2048 배렬에 필요한 수의 절반인 11 개의 주소선(A_0~A_{10})만이 존재한다는 것을 주목하자.

이것은 단자들의 수를 절약하기 위한것이다. 요구되는 22 개의 주소선들은 외부에 있는 선택론리를 통하여 소편에로 통과하며 11 개의 주소로 다중분배된다. 첫 11 개 주

소신호들은 배렬의 행주소를 지적하기 위하여 소편에 들어 가며 그다음의 다른 11 개 주소신호들은 렬주소로 된다. 이 신호들은 소편에 동기를 제공하기 위한 행주소선택(RAS)과 렬주소선택(CAS)에 의하여 동기화된다.

다중분배식주소화는 새롭게 갱신되는 모든 기억기소편들에서 4 배의 기억기크기를 가지게 하는 정 4 각형배렬의 리용을 추구한다. 주소가 하나 증가하면 행과 렬의 수가 각각 증가하므로 소편기억기의 크기는 4 배단위로 증가한다.

그림 4-4 는 재생회로가 포함되였음을 가리킨다. 모든 DRAM 들은 재생조작을 요구한다. 재생을 위한 단순한 기술은 실제상 모든 자료세포들이 재생되는 동안 DRAM 소편을 리용할수 없게 하는것이다. 재생계수기는 모든 렬들을 지적한다. 매개 렬을 재생하기 위하여 재생계수기로부터 나오는 출력선들은 렬해신기에 공급되며 RAS 신호가 능동으로 된다. 이것은 재생하려는 렬에 있는 모든 세포들을 재생시킨다.

4. 소편팩키지화

제 2 장에서 언급한바와 같이 직접회로는 다른 장치들에 련결하기 위한 단자들을 포함하는 팩키지에 설치된다.

그림 4-5 ㄱ는 1M×8 로 구성된 8Mbit 소편인 EPROM 팩키지실례를 주었다. 이 경우에 구성은 한단어 배렬팩키지로서 취급된다. 팩키지는 표준소편팩키지규격의 하나인 32개의 단자를 가지고 있다. 단자들은 다음과 같은 신호선들을 제공하고 있다. 32개의 단자를 가지고 있다.

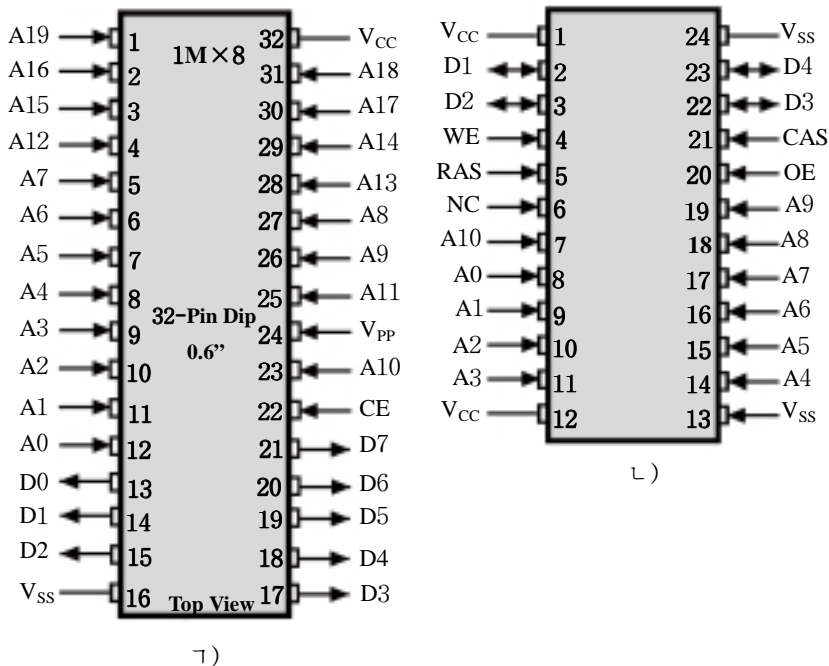



그림 4-5. 대표적인 기억소자단자와 신호
ㄱ-8Mbit EPROM, ㄴ-16Mbit RAM

- 단어호출을 위한 주소: 1M 단어에 대하여 총 20(2^{20}=1M)개의 단자가 요구된다. (A_0~A_{19})
- 8개의 선(D_0~D_7)으로 구성된 읽기를 위한 자료
- 소편에 공급하기 위한 전원(Vcc)
- 접지단자(Vss)
- 소편허가단자(CE). 매개 소편들이 같은 주소모선에 의하여 련결되는 여러개의 기억기소편들이 존재하므로 CE 단자는 지적되는 주소가 해당 소편에 대한 것인가 아닌가를 지적하는데 리용된다. CE 단자는 주소모선의 웃부분에 련결된 론리회로에 의하여 작용한다.
- 쓰기조작동안 공급되는 프로그람쓰기전압단자

대표적인 DRAM 단자구성은 4M × 4 로 구성된 16Mbit 소편으로 그림 4-5 ㄴ에 보여 주었다. ROM 소편과는 여러가지 차이점이 있다. RAM 은 갱신할수 있으므로 자료단자는 입력/출력기능을 가진다. 쓰기허가(WE)와 출구허가(OE)단자들은 쓰기조작인

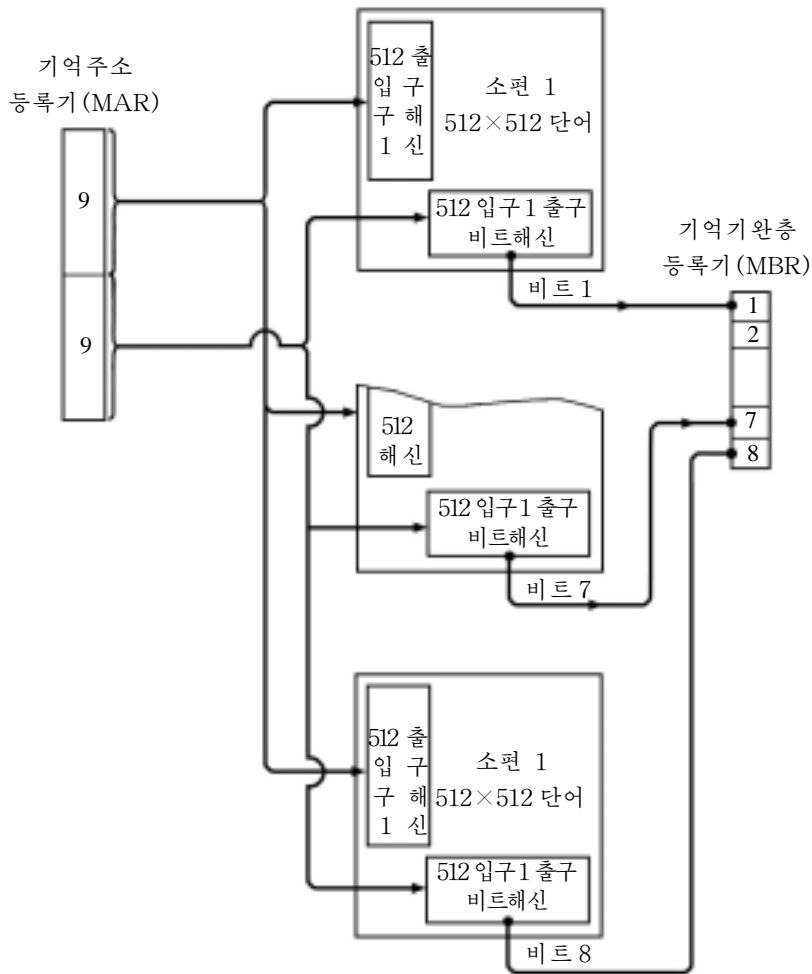



그림 4-6. 256Kbyte 기억기의 조직

가 읽기조작인가를 지적한다. DRAM 이 행과 렬에 의하여 호출되고 주소가 다중분배식이므로 4M 행/렬결합($2^{11} \times 2^{11}=2^{22}$)을 지적하는데 11개의 주소단자가 요구된다. RAS와 CAS 단자들의 기능은 앞에서 론의되였다.

5. 장치조직

RAM소편이 단어당 1bit를 포함한다면 이때 명백히 적어도 단어당 비트수와 같은 소편수가 요구될것이다. 실례로 그림 4-6에서는 256K × 8bit 단어로 이루어 지는 기억

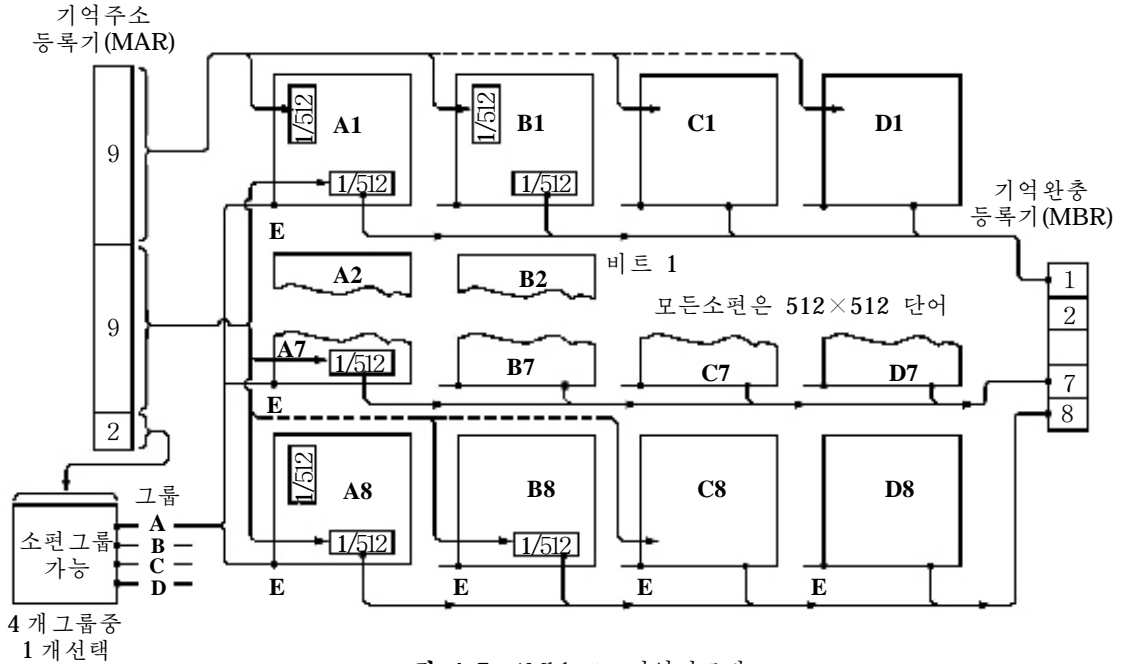



그림 4-7. 1Mbbyte 기억기조직

기장치를 어떻게 구성하는가를 보여 주었다. 256K 단어들에 대하여 18 bit 주소가 요구되며 일부 외부원천대상들로부터 장치에 주어 진다. 주소는 8개의 256K × 1bit 소편들에 제공된다.여기서 매개 소편들은 1bit 입력/출력선을 가진다. 이 구성은 기억기의 크기가 소편당 비트수와 같은 경우에만 실현할수 있다. 더 큰 기억용량이 요구되는 경우에 소편의 배렬들이 요구된다. 그림 4-7에서는 8bit 단어로서 1M 단어크기의 기억기에 대한 가능한 구성을 보여 주었다. 이 경우에 그림 4-6에서와 같이 배렬된 256K 단어를 포함하는 장치 4개로 구성한다. 1M 단어에 대하여 20개의 주소선이 요구된다. 아래의 18개의 비트들은 32개의 모든 소편들에 배선한다. 우의 2 bit는 4개의 장치묶음들중 한 장치에 소편허가신호를 보내는 그루빠선택론리장치에 입력시킨다.

6. 오유수정

반도체기억기체계는 오유문제들을 가진다. 이것은 파괴적인 오유와 비파괴적인오유들로 나눌수 있다. **파괴적인 오유**는 작용을 받은 기억기세포 혹은 세포들이 믿음성

있게 자료를 기억할수 없게 하는 영구적인 물리적고장이다. 즉 0 혹은 1 로 고정되거나 0 과 1 사이에 불안정하게 바뀐다. 이 오유는 매우 거치른 환경에서의 리용이나 제작상 오유, 장치의 피로에 의하여 발생한다. 비파괴적인 오유는 기억기파괴가 없이 하나 혹은 여러개의 기억세포들의 내용을 변화시키는 자유적이고 비파괴적인 사건에 의한 오유이다. 비파괴적인 오유는 전원공급문제들이나 미세한 α선알갱이에 의하여 발생한다. 이 미세한 α선알갱이는 일반적으로 모든 금속들가까이에서 적은 량으로 방사성알갱이들이 나타나므로 생기는 방사성부식으로부터 나타나며 이것으로 하여 기억기사용을 불편하게 한다. 이 파괴적인것과 비파괴적인 오유들은 시끄러우므로 대부분 현대주기억기들은 오유를 검사하고 수정하기 위한 론리회로들을 포함하고 있다.

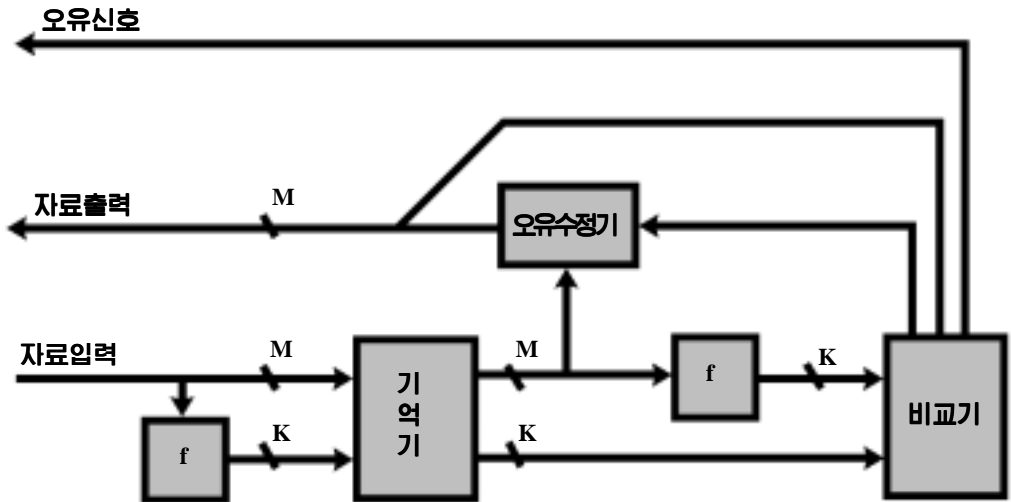



그림 4-8. 오유수정부호

그림 4-8 은 어떤 과정으로 오유수정이 진행되는가를 일반적인 과정으로 설명하였다. 자료가 기억기로부터 읽기될 때 함수 F 로 주어 지는 계산은 부호를 생성하기 위하여 자료에 대하여 수행된다. 부호와 자료는 기억된다. 따라서 M 비트단어의 자료가 기억되고 부호를 K 비트길이라고 하면 기억된 단어의 실제길이는 M+K 비트이다.

이전에 기억된 단어를 읽기할 때 부호는 오유를 검사하고 가능한껏 수정하는데 리용된다. K 개의 부호비트들의 새로운 묶음은 M 개의 자료비트들로부터 생성되며 꺼내여진 부호비트들과 비교된다.

비교결과는 다음의 세가지 결과중의 하나로 나타난다.

- 오유가 탐색되지 않았다. 꺼내여 진 자료는 전송된다.
- 오유가 발견되였고 오유수정이 가능하다. 자료비트들과 오유수정비트들을 묶어서 전송하기 위한 수정된 M 개의 비트묶음을 만드는 수정장치에로 들여 보낸다.

• 오유가 발견되였으나 수정할수 없다. 이 조건은 통보된다.

이러한 형태로 만든 부호를 **오유수정부호**라고 한다. 부호는 수정하고 검사할수 있는 단어의 오유비트수에 의하여 특징 지어 진다.

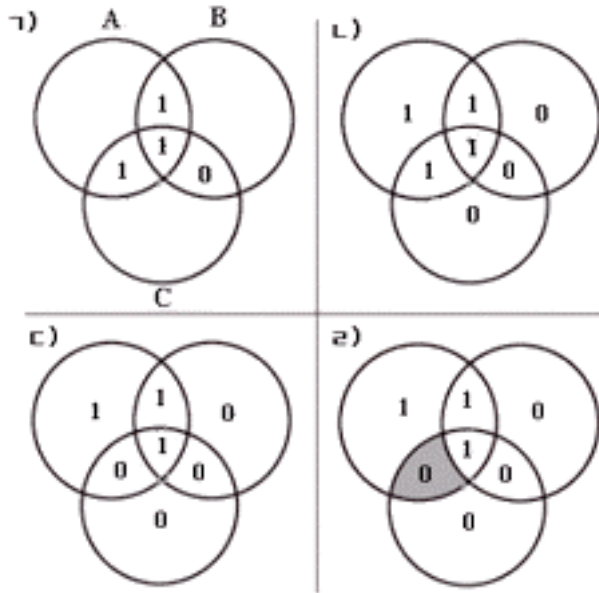



그림 4-9. 하밍오유수정부호

가장 간단한 오유수정부호는 Bell연구소의 리챠드 하밍에 의하여 제안된 **하밍부호**이다. 그림 4-9에서는 4bit단어들에 이 부호의 리용을 설명하기 위하여 벤의 그림풀이를 리용하였다. 3개의 겹쳐 있는 원들로 7개의 작은 부분을 만든다. 4개의 자료비트들은 안쪽의 작은 부분들에 배치한다(그림 4-9 ㄱ). 나머지 작은 부분들은 **기우성비트**들로 채워 넣는다. 매개 기우성비트들은 해당 원에서 1의 수가 우수가 되도록 값을 설정한다(그림 4-9 ㄴ). 따라서 원 A는 3개의 1을 포함하므로 이 원의 기우성비트는 1로 설정된다. 이제 오유로서 자료비트들중 하나가 변하였다면(그림 4-9 ㄷ) 쉽게 알수 있다. 기우성비트들을 검사하면 모순은 원 A와 원 C에서 발생하고 원 B에서는 발생하지 않았다. 반드시 7개의 작은 부분들중에 변한것은 A와 C에 있으며 B에는 없다. 그러므로 오유는 찾은 비트를 변경시켜 수정할수 있다.

포함되는 개념들을 명백히 하기 위하여 8BIT단어에서 한개의 비트오유를 찾고 수정할수 있는 부호를 개발하자.

먼저 얼마나 긴 부호가 있어야 하는가를 계산하자. 그림 4-8을 참고하면 비교론리는 2개의 K 비트값들을 입구로 받아 들인다. 비트와 비트사이 비교는 두 입구들의 안맞음론리합으로 진행된다. 그 결과를 신드롬단어라고 한다. 따라서 신드롬의 매 비트는 두 입구렬에 대하여 해당한 위치의 비트들이 같은가 같지 않은가에 따라 0 혹은 1로 된다. 그러므로 신드롬단어는 K개 비트폭으로 되며 0부터 2K-1사이의 값을 가진다.

값 0은 오유가 없다는것을 의미하며 나머지 2^K-1개의 값들은 만일 오유가 있다면 어느 비트에서 오유가 발생했다는것을 지적한다. 오유가 임의의 M개의 자료비트들 혹은 K개의 검사비트에서 발생하므로, $2^K-1 \geqq M+K$를 만족하여야 한다. 이 식은 M개의 자료비트들을 포함하고 있는 자료에서 한개의 비트오유를 수정하기 위하여 요구되는 비트들의 수를 정한다. 표 4-3에 여러 자료단어길이들에서 요구되는 검사비트들의 수를 제시하였다.

이 표로부터 8개 비트로 구성된 단어는 4개의 검사비트를 요구한다는것을 알수 있다. 편리를 위하여 다음의 특성을 가진 4bit 오유신드롬을 발생시키기로 하자.

표 4-3. 오유수정단어길이에서 증가

	단일오유수정		단일오유수정/ 2중오유검사	
자료비트수	검사비트수	증가률(%)	검사비트수	증가률(%)
8	4	50	5	62.5
16	5	31.25	6	37.5
32	6	18.75	7	21.875
64	7	10.94	8	12.5
128	8	6.25	9	7.03
256	9	3.52	10	3.91

- 신드롬이 0을 포함한다면 오유가 없다는것이다.
- 신드롬이 1로 설정된 비트 하나만을 포함하면 오유는 4개의 검사비트들중의 어느 하나에서 발생한 오유이므로 수정을 요구하지 않는다.
- 신드롬이 1로 설정된 비트가 둘이상이라면 신드롬의 수값은 오유가 발생한 자료비트의 위치를 지적한다. 이 자료비트는 수정을 위하여 반전된다.

이 특성을 얻기 위하여 자료와 검사비트들은 그림 4-10에서 보여 준바와 같이 12bit 단어로 배렬된다. 비트위치들을 1로부터 12까지 번호를 붙인다. 위치번호가 2의 제곱인 비트위치들은 검사비트로서 설계되였다. 검사비트들은 다음과 같이 계산되였다.

비트위치	1	2	3	4	5	6	7	8	9	10	11	12
	0	0	0	0	0	0	0	1	1	1	1	1
위　치	0	0	0	1	1	1	1	0	0	0	0	1
번　호	0	1	1	0	0	1	1	0	0	1	1	0
	1	0	1	0	1	0	1	0	1	0	1	0
검사비트	C1	C2		C4				C8				
자료비트			M1		M2	M3	M4		M5	M6	M7	M8

그림 4-10. 자료비트와 검사비트의 배렬

여기서 기호 ⊕는 안맞음론리합을 의미한다.

$$
\begin{aligned}
C1 &= M1 \oplus M2 \oplus M4 \oplus M5 \oplus M7 \\
C2 &= M1 \oplus M3 \oplus M4 \oplus M6 \oplus M7 \\
C4 &= M2 \oplus M3 \oplus M4 \oplus M8 \\
C8 &= M5 \oplus M6 \oplus M7 \oplus M8
\end{aligned}
$$

모든 검사비트들은 위치번호가 대응하는 렬위치에서 1 을 가지는 모든 자료비트위치에 대하여 연산한다. 따라서 자료비트위치들인 3, 5, 7, 9, 11 모두는 2^0 자리가 1 이며 비트위치들인 3, 6, 7, 10, 11 모두는 2^1 자리가 1 이며 비트위치 5, 6, 7, 12 는 2^2 자리가 1 이며 비트위치 9, 10, 11, 12 는 2^3 자리가 1 이다. 또 하나의 방법을 고찰하면 비트위치 n 은 $\sum i = n$ 인 C_i 비트들에 의하여 검사된다. 실례로 위치 7 은 위치 4, 2, 1 에서의 비트들에 의하여 검사된다. 즉 7 = 4 + 2 + 1.

이 방법을 실례를 통하여 고찰하자. 8bit 입구단어가 00111001 이라고 가정하자. 여기서 제일 오른쪽 위치가 M1 이다. 계산은 다음과 같다.

$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$
$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

이제 비트 3 이 오유로 되였으며 0 으로부터 1 로 변하였다고 가정하자. 검사비트가 다시 계산될 때 다음의 결과가 얻어 진다.

$$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 0 = 1$$
$$C2 = 1 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$C4 = 0 \oplus 1 \oplus 1 \oplus 0 = 0$$
$$C8 = 1 \oplus 1 \oplus 0 \oplus 0 = 0$$

새로운 검사비트들을 낡은 검사비트들과 비교할 때 신드롬단어는 다음과 같다.

	C8	C4	C2	C1
	0	1	1	1
⊕	0	0	0	1
	0	1	1	0

비트자리	12	11	10	9	8	7	6	5	4	3	2	1	
자료비트	M8	M7	M6	M5		M4	M3	M2		M1			
검사비트					C8				C4		C2	C1	
	1	1	1	1		0	0	0		0			C80
	1	0	0	0		1	1	1		0			C41
	0	1	1	0		1	1	0		1			C21
	0	1	0	1		1	0	1		1			C11
단어기억	1	0	1	1	0	1	0	0	1	1	1	1	
단어꺼내기	0	0	1	1	0	1	1	0	1	1	1	1	
	1	1	1	1		0	0	0		0			C80
	1	0	0	0		1	1	1		0			C40
	0	1	1	0		1	1	0		1			C20
	0	1	0	1		1	0	1		1			C11

그림 4-11. 검사비트변경

결과는 0110 이며 자료비트 3 을 가리키는 비트위치 6 이 오유로 나타났다는것을 지적한다.

그림 4-11 에서는 우의 계산과정을 설명하였다. 자료와 검사비트들은 12bit 단어로 적합하게 배치하였다. 렬에 매 자료비트들의 위치번호를 설정함으로써 매 행에서의 1 들이 그 행에 대한 검사비트에 의하여 검사되는 자료비트들을 가리킨다. 결과3은 1에 의해서만 영향을 받으므로 1 을 포함하는 렬들만 구분하기 위하여 동그라미로 표시한 다음 검사비트들은 행별로 계산된다. 결과는 원래 자료비트들에 대해서와 오유를 포함하는 자료비트들에 대하여 보여 주었다.

이렇게 얻어 진 부호를 **단일오유수정**(SEC)부호라고 한다. 일반적으로 반도체기억기는 단일오유수정과 2중오유탐색(SEC-DED)부호로 설치되였다. 표 4-3이 보여 주는 것처럼 이와 같은 부호들은 SEC 부호와의 비교에 추가적인 비트 하나를 더 요구한다.

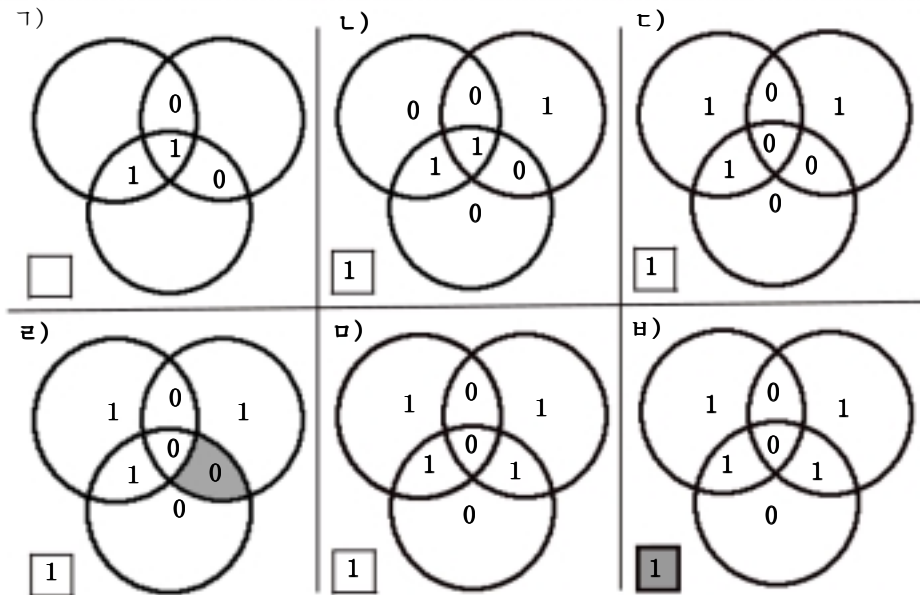



그림 4-12. 하밍 SEC-DEC 부호

그림 4-12에서는 4bit 자료단어로 주어 졌을 때 SEC-DED부호를 어떻게 작성하는가를 보여 주었다. 순서는 두개 오유가 발생하였을 때(그림 4-12 ㄷ) 검사순서는 방향을 잡지 못하며 (그림 4-12 ㄹ) 세번째 오유가 발생하였을 때 문제는 어렵게 된다는것을 보여 준다(그림 4-12 ㅁ). 이 문제를 극복하기 위하여 총 1 의 개수가 우수가 되도록 8번째 비트로 설정하여 추가한다. 추가된 기우성비트는 오유를 발견한다.

오유수정부호는 복잡성을 추구하지만 기억기의 믿음성을 높여 준다. 한비트 소편구성에서는 일반적으로 SEC-DED가 적합하다고 고찰된다. 실례로 IBM 30xx 들을 주기억기에서 64bit 자료 매개에 대하여 8bit SEC-DED 를 리용한다. 따라서 주기억기의 크기는 실제로 사용자가 리용하는것보다 약 12%정도 더 크다. VAX 콤퓨터들은 기억기의 개별적인 32bit 자료에 대하여 7bitSEC-DED 를 리용하였으며 따라서 주기억기는 22%정도 더 커진다. 현대 DRAM들은 128bit자료에 대하여 9개의 검사비트를 리용하므로 기억기는 사용자가 리용하는것보다 7%정도 더 커진다.

제 3 절. 캐쉬

1. 원 리

캐쉬(고속완충기억기)는 주기억기를 가장 빠른 기억기들에 가까운 속도로 보장하기 위하여 그리고 동시에 반도체기억기들의 값 눅은 형들로 큰 용량의 기억기를 제공하기 위하여 리용된다. 개념은 그림 4-13에 주었다.

용량이 작고 속도가 빠른 기억기와 상대적으로 용량이 크고 속도가 뜬 기억기가 있다. 캐쉬는 주기억기의 부분적인 복사를 진행한다. 처리장치가 기억기의 단어를 읽으려고 시도할 때 그 단어가 캐쉬에 있는가하는 검사를 진행한다. 만일 있다면 단어는 처리장치에로 입구된다. 만일 없다면 고정된 길이의 단어들로 구성된 주기억기의 블로크가 캐쉬로 넣어 지며 그 다음 처리장치에 입력된다. 국부성참조현상에 의하여 자료의 블로크가 한번의 기억기참조로 캐쉬에 꺼내여 지면 그 블로크에 있는 다른 단어들은 앞으로 참조될 가능성이 충분하다.

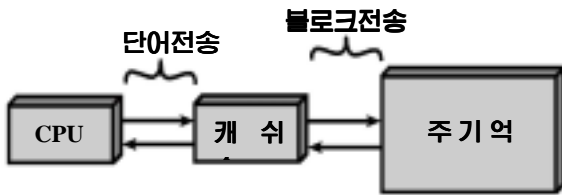



그림 4-13. 캐쉬와 주기억기

그림 4-14 에는 캐쉬/주기억기체계의 구조를 보여 주었다. 주기억기는 2^n 주소가능한 단어들로 구성되며 모든 단어들은 유일하게 n 비트주소를 가진다. 목적달성

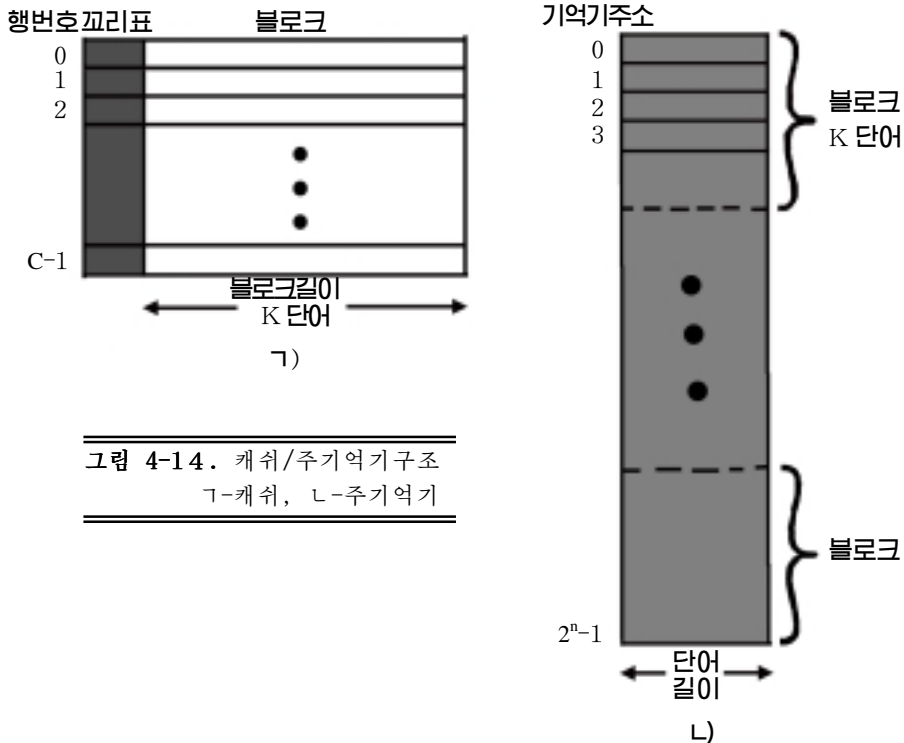



그림 4-14. 캐쉬/주기억기구조
ㄱ-캐쉬, ㄴ-주기억기

을 위하여 이 기억기는 매개가 K 단어로 고정된 블로크를 여러개 가지고 있다고 하자. 즉 $M=2^n/K$ 개의 블로크를 가진다. 캐쉬는 K 개 단어단위의 C 개 행들로 구성되며 이 행의 개수는 주기억블로크의 수보다 작다($C < M$).

임의의 순간에 기억기의 블로크들의 어떤 작은 묶음은 캐쉬의 행들에 상주한다. 만일 기억기의 어떤 블로크의 단어가 읽기되였다면 그 블로크는 캐쉬의 행들중의 어느 하나에로 전송된다. 캐쉬의 행수보다 기억기의 블로크수가 더 많으므로 개별적인 행들은 유일적으로, 영구적으로 특정의 블로크들에 할당될수 없다. 따라서 매개 행들은 특정의 블로크가 현재 기억되여 있다는것을 지적하는 표식표(tag)를 가진다. 표식자는 보통 주기억주소의 한 부분이다.

그림 4-15 에 읽기조작을 주었다. 처리장치는 주소와 읽으려는 단어의 RA 를 내보낸다. 단어가 캐쉬에 있으면 처리장치로 끌어 들인다. 그렇지 않다면 그 단어를 포함하고 있는 블로크는 캐쉬로 넣기되며 호출된 단어는 동시에 처리장치로 들어 간다. 그림 4-15 는 이 마지막 두 조작이 병렬로 진행된다는것을 보여 주며 현재 캐쉬조직의 표준인 그림 4-16 에서 보여 준 구성은 이것을 반영하고 있다. 이 구성에서 캐쉬는 자료, 조종, 주소선들로 처리장치에 련결된다. 또한 자료와 주소선은 주기억기가 결합되여 있는 체계모선에 추가된 자료와 주소완충기들에 결합된다. 캐쉬명중이 발생하면 자료와 주소완충기는 리용되지 않으며 통신은 체계모선을 통과하지 않고 처리장치와 캐쉬사이에서만 진행된다. 캐쉬실패가 발생하면 요구되는 주소는 체계모선으로 전송되며 자료는 캐쉬와 주기억기로 자료완충기를 통하여 복귀된다. 다른 구성들에서 캐쉬는 모든 자료, 주소, 조종선들에 대하여 처리장치와 주기억장치사이에 물리적으로 삽입된다. 이 경우 캐쉬실패에 대하여 캐쉬리용과 관련되는 성능파라메터들에 대한 론의는 부록 4-1 에 주었다.

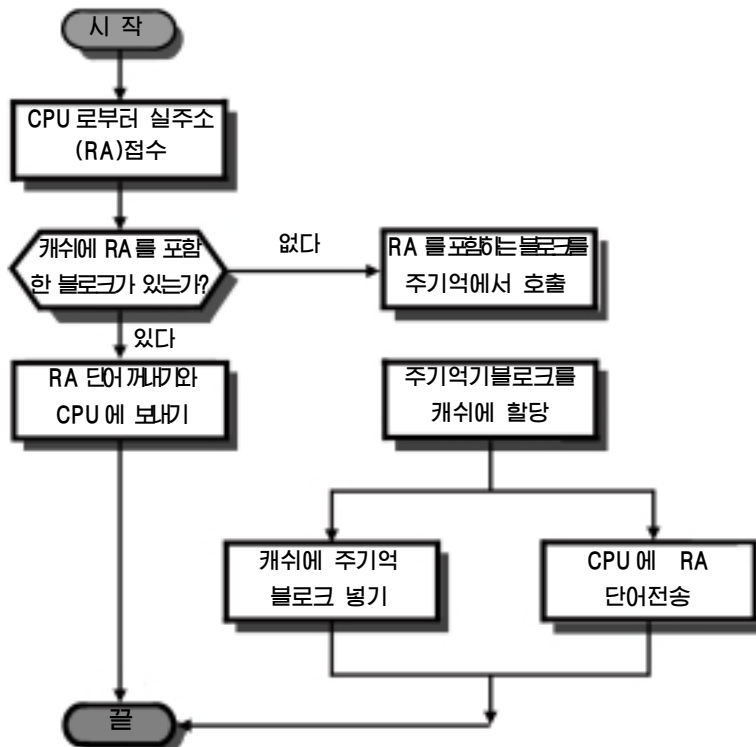



그림 4-15. 캐쉬읽기조

2. 캐쉬설계의 요소

캐쉬실체들에는 종류가 많지만 캐쉬방식을 분류하고 구별하는데 봉사하는 기본설계요소들은 적다. 표 4-4 에 기본요소들을 제시하였다.

캐쉬크기

첫 요소로서 캐쉬크기는 이미 론의되였다. 비트당 전체 평균가격이 주기억기만 리용할 때와 비슷할 정도로 크게 캐쉬의 크기를 정하면 된다. 최소 캐쉬크기에 대한 여러가지 다른 류형들이 있다. 캐쉬가 크면 캐쉬주소화에 포함되는 론리문수도 커진다. 결과 큰 캐쉬들은 작은 캐쉬들보다 속도가 약간 떠지게 되는 경향성을 가지며 같은 집적회로기술로 실현하고 소편과 인쇄기판의 같은 위치에 배치되는 경우도 마찬가지이다. 캐쉬의 크기는 리용가능한 소편과 인쇄기판면적에 의하여 영향을 받는다. 부록 4-1 에 제시된바와 같이 여러 연구들에서는 1K 와 512K 단어사이의 캐쉬크기가 효과적이라는것을 시사하고 있다. 캐쉬의 성능이 작업부하의 내용에 아주 민감하므로 하나의 적당한 캐쉬의 크기로 하는것은 불합리하다.

표 4-4. 캐쉬설계의 요소

캐쉬크기	**쓰기방법**
배치기능	동시쓰기
직접	비동시쓰기
완전련상	한번쓰기
묶음련상	**행크기**
치환알고리듬	**캐쉬수**
LRU	단일 혹은 2 준위
FIFO	동일 혹은 분할
LFU	
자유	

배치기능

주기억기의 블로크수보다 캐쉬행들은 더 작으므로 캐쉬행에 주기억블로크들을 배치하기 위한 알고리듬이 요구된다. 앞으로 이 방법들은 주기억블로크가 현재 캐쉬행에 있는가를 검사하는데 요구된다. 배치기능의 선택은 캐쉬가 어떻게 구성되였는가에 달려 있다. 직접배치, 완전련상배치, 묶음련상배치와 같은 세가지 기술이 리용될수 있다. 이 동작원리들에 대하여 설명하자. 매 경우에 일반적인 구조와 주어 진 실례를 고찰한다. 세가지 경우들에 대하여 실례는 다음의 요소들을 포함한다.

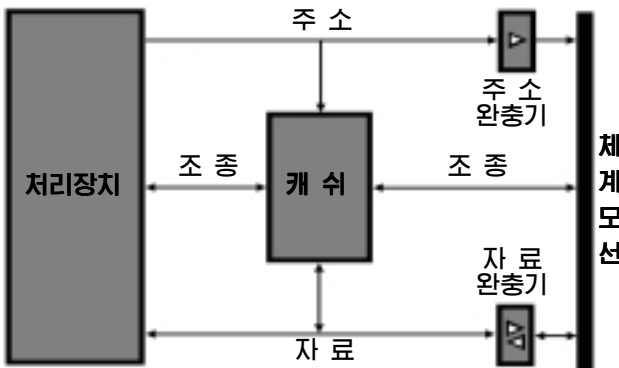



그림 4-16. 일반적캐쉬조직

- 캐쉬는 64KB 를 가진다.
- 자료는 주기억기와 캐쉬사이에서 4byte 의 블로크로 전송된다. 이것은 캐쉬가 $16K=2^{14}$ 개의 4byte 행들로 구성되였다는것을 의미한다.

- 주기억기는 16Mbyte 로 구성되였으며 모든 바이트들은 24bit 주소(2^{24}=16M)로 직접 주소화가 가능하다. 따라서 배치목적을 위하여 주기억기는 4M 개의 4byte 블로크로 구성되였다고 고찰할수 있다.

직접배치방식이라고 하는 가장 단순한 기술은 주기억기의 매개 블로크들이 오직 주어 진 한개의 가능한 캐쉬행에 배치되도록 한다. 그림 4-17 에서는 일반적인 구성을 보여 주었다. 배치는 다음과 같이 표시된다.

$$i \;=\; j \;\; \text{modulo} \;\; m$$

여기서 i - 캐쉬행번호,
j - 주기억기의 블로크번호,
m - 캐쉬에서 행들의 수

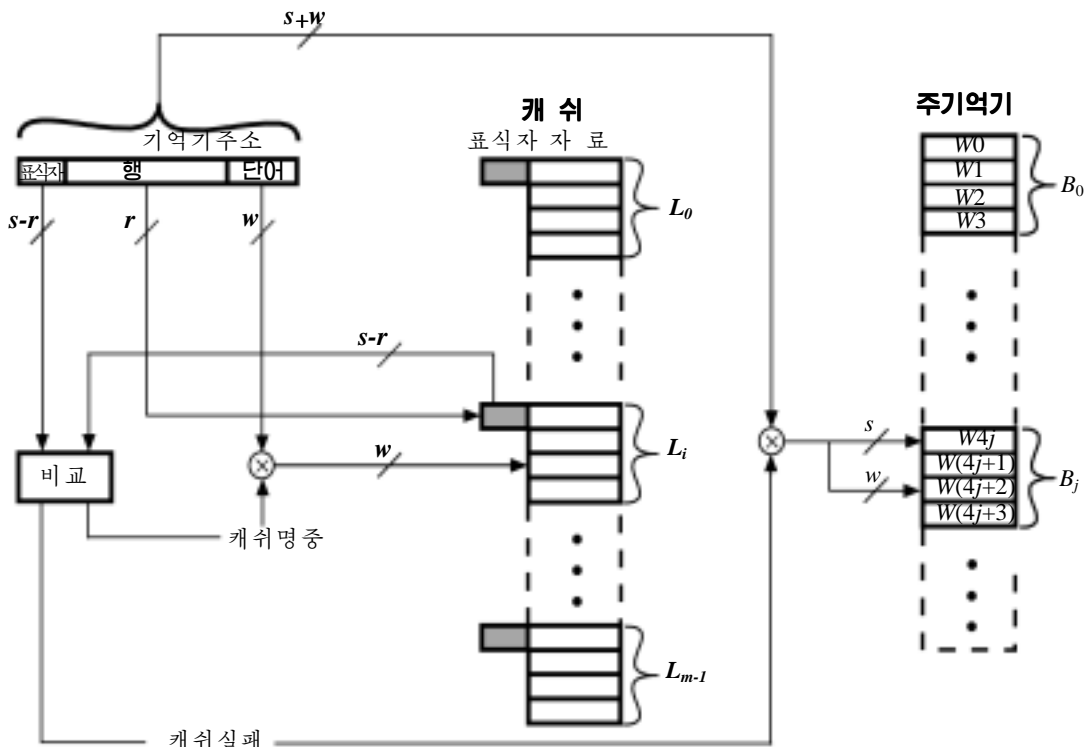



그림 4-17. 직접배치캐쉬구성

배치기술은 주소를 리용하여 쉽게 실현된다. 캐쉬호출목적을 위하여 개별적인 주기억주소들은 3 개 마당의 구성으로 볼수 있다. 제일 마지막의 w 개 비트들은 주기억기블로크안에서 유일하게 단어 혹은 바이트인가를 지적한다. 즉 대부분 현대콤퓨터들에서 주소는 바이트준위이다. 나머지 s 개의 비트들은 주기억기의 2^s 개의 블로크중 하나를 지적한다. 캐쉬론리회로는 s-r 비트(제일 웃 부분)개의 표식표와 r 개 비트의 행마당으로 분할한다. 이 마지막마당은 캐쉬의 m=2^K 개의 행들중의 하나를 가리킨다. 이 배치방식의 의미는 주기억블로크들이 다음과 같이 캐쉬행들에 배치된다는것이다.

캐쉬행	배치된 주기억 블로크
0	0, m, $2m$, … , 2^s-m
1	1, $m+1$, $2m+1$, … , 2^s-m+1
⋮	⋮
$m-1$	m-1, $2m-1$, $3m-1$, … , 2^s-1

따라서 행번호로서 주소부분의 리용은 캐쉬에 주기억기의 개별적인 블로크에 대한 유일한 배치를 제공한다. 블로크가 실제적으로 배치된 행에서 읽어 질 때 그 행에 대응하는 다른 블로크들로부터 구별하기 위한 자료를 덧붙이는것이 필요하다. 가장 웃부분의 $s-r$개 비트들은 이러한 목적에 리용된다.

그림 4-18 에서는 직접배치방식을 리용한 실례를 보여 주었다. 실례에서 m=16K=2^{14}, 그리고 $i = j$ modulo 2^{14}이다. 배치는 다음과 같이 된다.

캐쉬행	블로크의 시작 기억주소
0	000000, 010000, … , FF0000
1	000004, 010004, … , FF0004
⋮	⋮
$m-1$	00FFFC, 1FFFC, …, FFFFFC

같은 행번호로 배치되는 2 개의 블로크가 같은 표식자번호를 가지는 경우는 없다. 따라서 블로크 000000, 010000, FF0000 는 각각 표식표번호 00, 01, …FF 을 가진다.

그림 4-15 를 다시 고찰하면 읽기조작은 다음과 같이 진행된다. 캐쉬체계는 24bit 주소로 존재한다. 14bit 행번호는 캐쉬에서 특정의 행을 호출하기 위한 첨수로 리용된다. 만일 8bit 표식자번호가 현재 그 행에 기억된 표식표번호와 같다면 2bit 단어번호가 지적된 행에서 4byte 중의 하나를 선택하는데 리용된다. 같지 않다면 22bit 표식자와 행번호마당은 주기억기로부터 블로크를 꺼내는데 리용된다. 꺼내기에 리용되는 실제주소는 4byte 가 블로크경계의 시작부터 꺼내여 지도록 22bit 표식표와 행번호마당에 단어마당 두비트를 0 으로 하여 결합한다.

직접배치기술은 실현하기 간단하며 비용도 적게 든다. 기본결함은 임의의 주어 진 블로크에 대한 캐쉬의 위치가 고정된것이다. 따라서 프로그람이 같은 행에 배치되는 두개의 서로 다른 블로크들로부터 반복적으로 단어들을 참조하게 되였다면 이때 블로크들을 련속적으로 캐쉬에 번갈아 넣기되며 명중률이 낮아 진다.

련상배치방식은 주기억블로크들이 캐쉬의 임의의 행에 넣기되도록 허락함으로써 직접배치의 결함을 극복한 방식이다. 이 경우에 캐쉬조종론리는 표식자와 단어마당으로서 기억기주소를 간단히 갈라서 해석한다. 표식자마당은 유일하게 주기억기의 블로크를 지적한다.

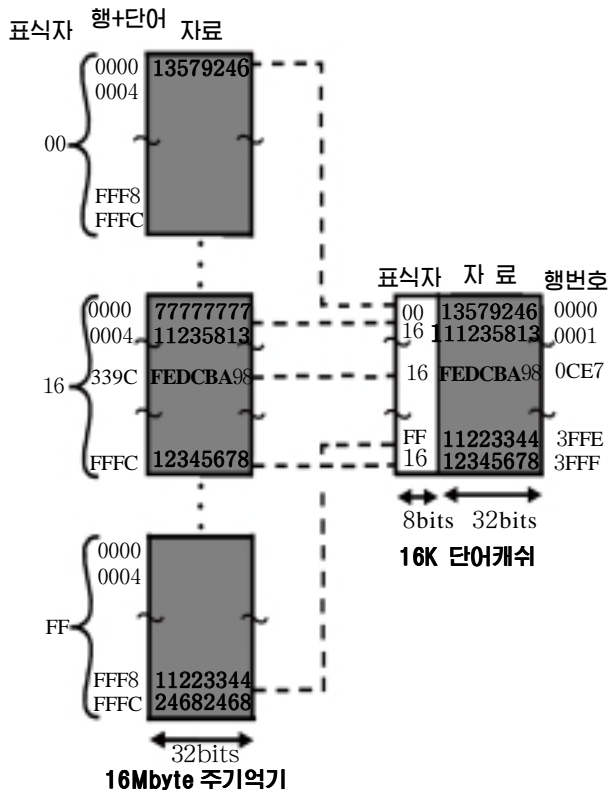



그림 4-18. 직접배치실례

블로크가 캐쉬에 있는가 없는가를 확정하기 위하여 캐쉬조종론리는 련속적으로 모든 행들의 표식자를 검사하여야 한다. 그림 4-19 에 이 조종론리를 주었다.

그림 4-20 은 련상배치방식을 리용한 실례이다. 주기억기주소는 22bit 표식자와 2bit 바이트번호로 구성된다. 22bit 표식자는 캐쉬에서 매개 행에 대응하는 자료의 32bit 블로크를 지적하여야 한다. 주소의 제일 왼쪽 22bit 는 표식자를 형성한다. 따라서 24bit 16 진수 주소 16339C 는 22bit 표식자 058CE7 을 가진다. 이것은 쉽게 2 진수표시법으로 볼수 있다.

주기억기주소	0001	0110	0011	0011	1001	1100	(2 진수)
	1	6	3	3	9	C	(16 진수)
표 식 표	00	0101	1000	1100	1110	0111	(2 진수)
	0	5	8	C	E	7	(16 진수)

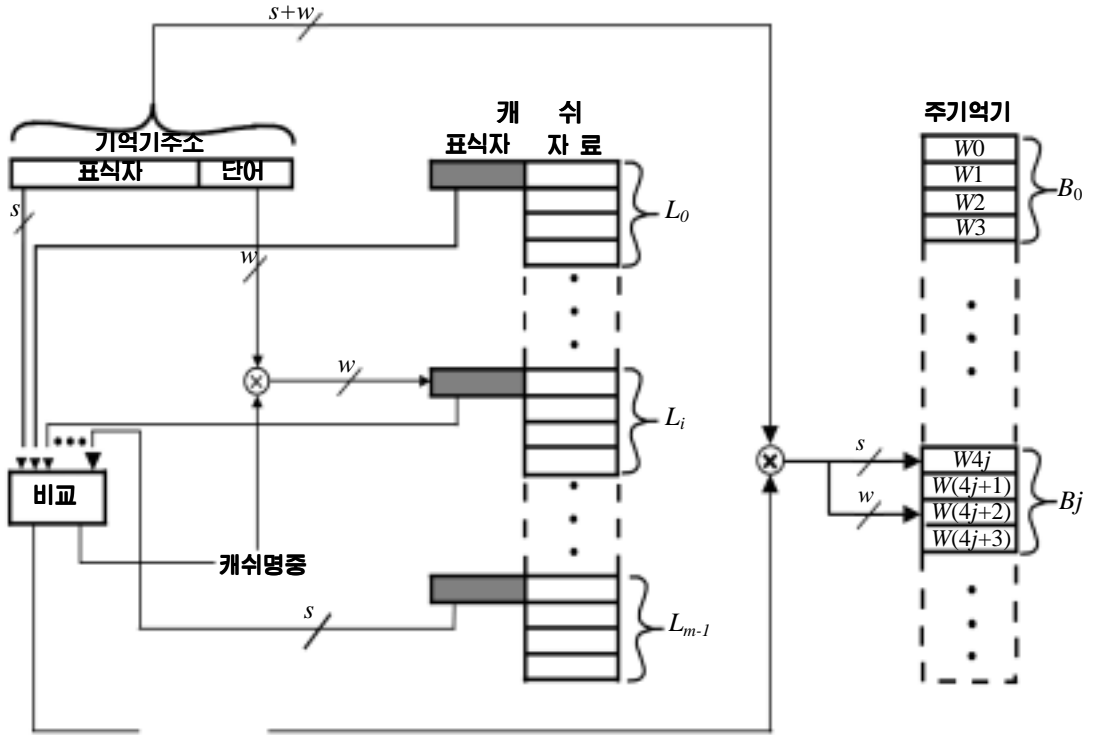



그림 4-19. 완전련상캐쉬조직

련상배치방식에서는 새로운 블로크가 캐쉬로 읽기될 때 임의의 블로크를 치환할수 있으므로 유연하다. 이 절의 뒤에서 론의하는 치환알고리듬들은 명중률을 최대로 하도록 설계되였다. 련상배치방식의 기본결함은 모든 캐쉬행들의 표식자를 병렬로 조사하기 위한 복잡한 회로를 가지는것이다.

묶음련상배치방식은 직접배치와 완전련상배치방법들의 우점을 살리는 한편 이 두 방식의 결함을 감소시킨 방식이다. 이 경우에 캐쉬는 k 개의 행들로 구성되는 ν 개의 묶음들로 분할된다. 관계식은 다음과 같다.

$$m = \nu \times k$$
$$i = j \text{ modulo } \nu$$

여기서 i - 캐쉬묶음번호,
j - 주기억블로크번호,
m - 캐쉬의 행수

이것을 k 통로 묶음련상배치방식이라고 한다. 묶음련상배치에서 블로크 B_j는 i번째 묶음의 임의의 행들에 배치될수 있다. 이 경우에 캐쉬조종론리는 기억기주소를 표식자, 묶음, 단어 이렇게 단순히 3 개의 마당으로 해석한다. d 개의 묶음비트들은 $\nu=2^d$ 개의 묶음중의 하나를 지적한다. s 개의 표식자와 묶음마당의 비트들은 주기억의 2^s 개의 블로크중 하나를 지적한다. 그림 4-21 에서는 캐쉬조종론리를 설명하였다. 완전련상배치에서는 기억기주소에서 표식자는 매우 크며 그 캐쉬안의 모든 행들의 표식자와 다 비교하여야 한다. k 통로묶음 련상배치방식에서는 기억기주소에서의 표식자가 매우 작으

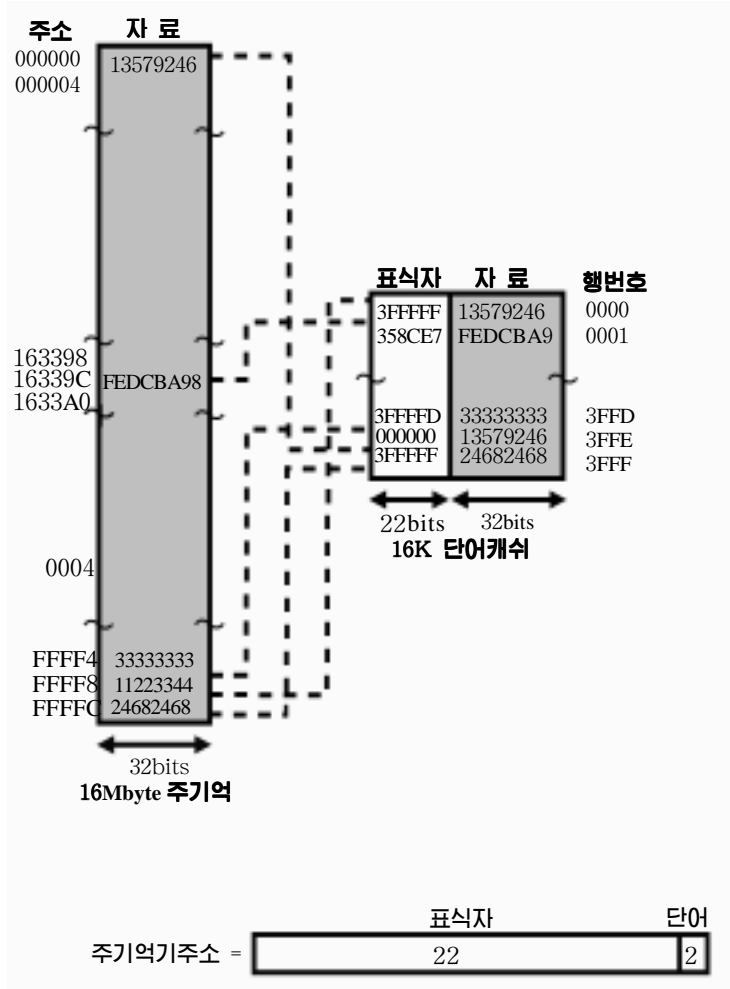



그림 4-20. 련상배치실례

며 한개의 묶음안에 있는 k 개의 표식자들에 대해서만 비교하면 된다.

그림 4-22 에서는 2 통로 묶음련상이라고 할수 있는 묶음당 2 개의 행으로 된 묶음련상배치방식을 리용한 실례를 보여 주었다. 13bit 묶음번호는 캐쉬안에서 두개의 행들로 된 한개의 묶음을 지적한다. 또한 주기억기의 블로크번호를 2^{13} 으로 나눈 나머지가 주어 진다. 이 나머지는 행들에 블로크들의 배치를 결정한다. 즉 주기억기의 블로크들인 00000,00A000,⋯,FF1000 는 캐쉬묶음 0 에 배치된다. 임의의 이 블로크들은 묶음에서 두개의 행들중 임의의 위치에 넣기될수 있다.

같은 캐쉬묶음에 배치된 두개의 블로크는 같은 표식자번호를 가지지 않는다는것에 주목해야 한다. 읽기조작에서 13bit 묶음번호는 어느 묶음이 조사되는가를 결정하는데 리용된다. 묶음의 두개 행들은 호출되는 주소의 표식자번호와 맞는가를 위하여 조사된다.

ν=m, k =1 인 경우에 묶음련상배치방식은 직접배치방식으로 되며 ν=1, k =m 인 경우에는 완전련상배치방식으로 된다. 묶음당 두 행(ν=m/2, k = 2)방식의 리용은 가

장 일반적인 묶음련상방식이다. 이 방식은 직접배치방식보다 명중률이 완전히 개선된

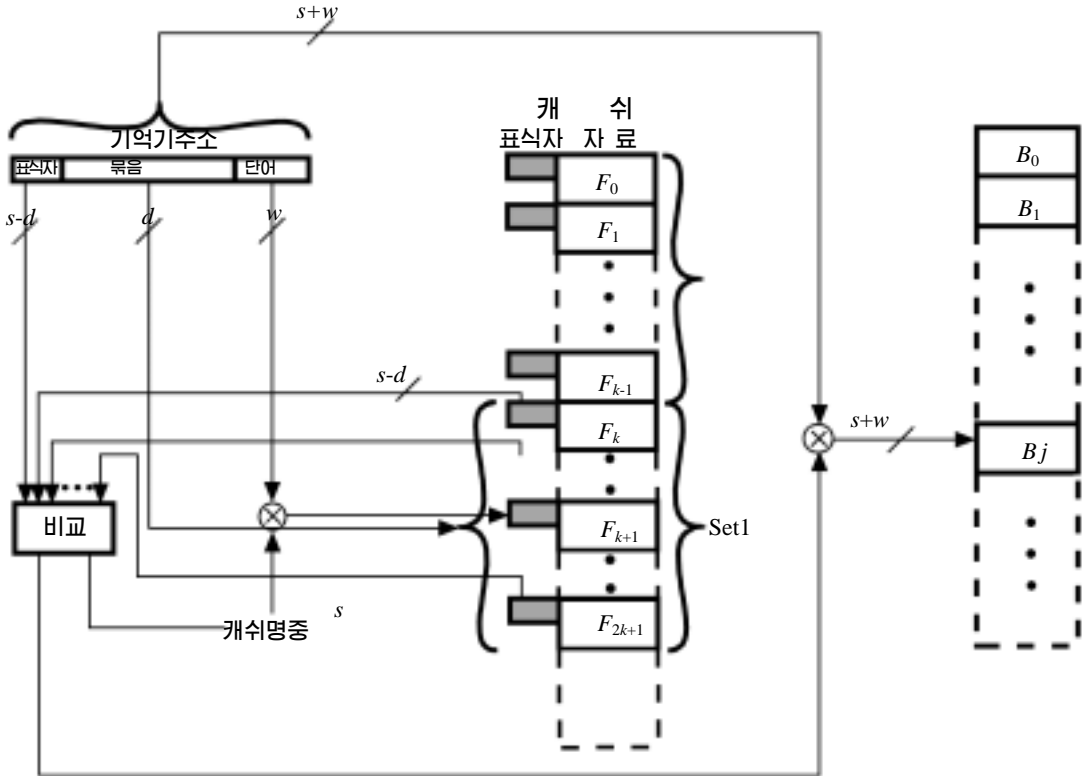



그림 4-21. 2 통로묶음련상캐쉬조직

다. 4 통로묶음련상(ν=m/4, k = 4)은 상대적으로 적은 추가적인 원가로 적당한 추가성능개선을 가진다. 묶음당 행수를 더 증가시키면 적은 효률을 가지게 된다.

치환알고리듬

새로운 블로크가 캐쉬로 들어 올 때 캐쉬에 존재하고 있는 블로크들중의 하나가 치환된다. 직접배치에서는 임의의 주어 진 블로크에 대하여 치환가능한 행하나가 고정적으로 주어 진다. 즉 치환가능한 후보선택이 없다. 완전련상과 묶음련상기술에서는 치환알고리듬이 요구된다. 치환의 속도를 높이기 위하여 이러한 알고리듬들은 장치적으로 실현되여야 한다. 여러가지 알고리듬들이 시도되고 있다. 가장 공통적인 네가지 알고리듬에 대하여 고찰한다. 대체로 가장 효률적인것은 LRU 방식이다. 이 방식은 가장 오래동안 참조되지 않으면서 캐쉬에 존재하는 블로크를 치환하는 방식이다. 두 통로묶음련상에서 이것은 아주 쉽게 실현된다. 매 행은 USE 비트를 포함한다. 행이 참조될 때마다 USE 비트는 1로 설정되며 묶음에서 다른 행의 USE 비트들은 0 으로 된다. 블로크가 묶음에로 읽어 질 때마다 USE 비트가 0 인 행을 리용한다. 시간적으로 가까이에 참조된 블로크가 참조될 가능성이 높다고 가정하므로 LRU 는 가장 높은 명중률을 가진다. 또 하나의 가능성은 선입선출(FIFO)방식이다. 이 방식은 캐쉬에 가장 오래 존재한 묶음의 블로크를 치환한

다. FIFO는 순환대기렬완충기기술로서 쉽게 실현한다. 또 하나의 가능성은 LFU방식이다. 이 방식은 가장 적은 참조를 예측하여 선정된 묶음의 블로크를 치환한다. LFU는 매 행에 계수기를 련상시킬 때에 실현할수 있다. 사용량에 기초하지 않는 기술은 후보행들중에서 우연히 한행을 포착하는것이다. 모의학습들은 자유치환이 습관상 기준으로 되는 알고리듬에 약간 뒤떨어 진 성능을 제공한다는것을 보여 준다.

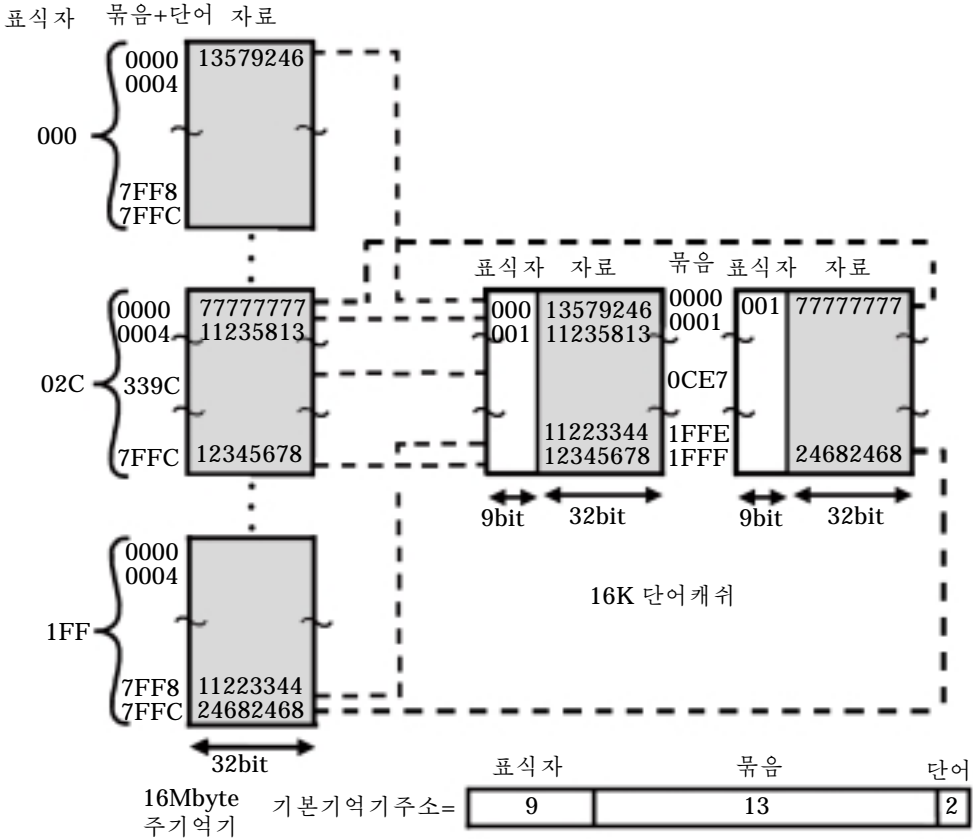



그림 4-22. 두통로묶음련상배치실례

쓰기방책

캐쉬에 들어 있는 블로크를 치환하기전에 그 블로크가 캐쉬안에서 변하였지만 대응하는 주기억에는 그렇지 못할수 있다는것을 고찰할 필요가 있다. 만일 주기억블로크가 변하지 않았다면 캐쉬에 있는 낡은 블로크를 주기억에 대하여 쓰기를 진행하여야 한다. 만일 주기억블로크가 변하였다면 적어도 한번의 쓰기조작이 캐쉬의 주어 진 행의 단어에 대하여 수행되며 이때 주기억기도 같이 갱신되는 방법을 리용한것이다. 성능과 실용적인 상품화지표들에 따라 여러가지 쓰기방법들이 가능하다. 경쟁상 두가지 문제가 있다. 첫째로는 하나이상의 장치들이 주기억기에 대한 호출을 진행하는것이다. 실례로 I/O 장치가 기억기에 직접 읽기/쓰기할수 있다. 어떤 단어가 캐쉬에서만 변하였다면 이때 대응하는 기억기단어는 무효로 된다. I/O장치가 주기억기를 변경시켰다면

이때 캐쉬단어가 무효로 된다. 여러개의 처리장치들이 같은 모선에 결합되고 매 처리장치들이 자기의 국부캐쉬를 가질 때 더 복잡한 문제들이 발생한다. 이때 어떤 단어가 캐쉬에서 변하였다면 다른 캐쉬들에 있는 단어들은 무효화될수 있다.

가장 간단한 기술은 **동시쓰기방식**이다. 이 기술을 리용하면 모든 쓰기조작은 캐쉬와 동시에 주기억기에 대하여 진행되며 주기억기가 항상 유효하게 보존된다. 어떤 다른 처리장치- 캐쉬장치는 그 캐쉬안에서 일치성을 유지하도록 주기억에 대한 입출력관계를 감시할수 있다. 이 기술의 기본결함은 실제적인 기억기입출력관계가 발생하므로 병목현상(성능제한현상)이 발생한다는것이다. 비동시쓰기방식이라고 하는 또 하나의 기술은 기억기쓰기를 최소로 하게 한다. 비동시쓰기에서는 캐쉬에 대해서만 갱신을 진행한다. 갱신조작이 일어 나면 행과 관련된 UPDATE 비트가 설정된다. 그다음 블로크가 치환될 때 UPDATE 비트가 설정되여 있으면 해당 주기억기위치가 무효화되였다는것이며 결국 I/O 장치에 의한 호출은 캐쉬를 통해서만 진행될수 있다. 이것은 복잡한 회로구성과 병목현상을 초래한다. 경험은 쓰기조작에 의한 기억기호출비률이 15%정도라는것을 보여 준다.

캐쉬와 공유기억기를 가진 하나이상의 장치(대표적으로 처리장치)를 가진 모선구성에서 새로운 문제점들이 나타난다. 어떤 하나의 캐쉬에서 자료가 변하면 이것은 주기억기의 대응하는 단어뿐아니라 다른 캐쉬와 같은 단어도 무효화한다(만일 어떤 다른 캐쉬가 같은 단어를 가지는 경우가 발생한다면). 만일 동시쓰기방법이 리용되였다면 다른 캐쉬는 무효자료를 포함한다. 이 문제를 방지하는 체계는 캐쉬의 일치성을 유지한다고 말한다. 캐쉬일치성을 위한 가능한 방법들은 다음과 같다.

- **동시쓰기감시모선방법**: 매개의 캐쉬조종기들은 다른 모선조종기들에 의한 기억기쓰기조작을 검사하기 위하여 주소모선을 감시한다. 만일 다른 조종장치가 캐쉬기억기에 들어 와 있는 공유기억기위치에 대한 쓰기를 진행하였다면 캐쉬조종기는 캐쉬의 해당한 부분을 무효화한다. 이 방법은 모든 캐쉬조종장치들에 대한 동시쓰기방법의 리용에 의존한다.
- **장치적전면복사방식**: 캐쉬를 통한 주기억기의 모든 갱신이 모든 캐쉬들에 복사하게 하는것을 담보하는 추가적인 장치가 리용된다. 따라서 어떤 처리장치가 자기 캐쉬의 단어를 변경시켰다면 이 갱신은 주기억기에 써지며 동시에 다른 캐쉬들에서 대응되는 단어들도 같이 갱신된다.
- **무캐쉬기억기방식**: 주기억기의 일부분만이 하나이상의 처리장치들에 공유되며 이 부분은 캐쉬가 없이 설계되였다. 이와 같은 체계에서 공유기억기의 모든 호출은 공유기억기가 캐쉬에로 복사되지 않으므로 캐쉬실패를 가진다. 무캐쉬기억기는 소편선택론리나 높은 주소비트들을 리용하여 지적할수 있다.

캐쉬일치성은 활성화된 연구분야이다. 이 문제는 제 16 장에서 더 전개한다.

행크기

또 하나의 설계요소는 행의 크기이다. 자료의 블로크가 캐쉬에 들어 와 배치될 때 요구되는 단어뿐 만아니라 린접한 몇개의 단어들로 입력된다. 블로크크기가 매우 작은것으로부터 증가하면 초기에는 참조된 자료의 근방에 있는 자료들은 가까운 시기에 다시 참조될수 있다는 국부성의 법칙에 의하여 캐쉬명중률이 증가한다. 계속 블로크의

크기가 증가하면 더 효률적인 자료가 캐쉬에로 들어 오게 된다. 그러나 블로크가 너무 커지면 새롭게 꺼내여 진 정보의 리용부분은 치환되는 정보의 재리용부분보다 적어 지므로 캐쉬명중률은 감소된다. 따라서 두가지 특수영향을 받는다.

- 큰 블로크는 캐쉬에 넣어 지는 블로크들의 수를 적게 한다. 매개 블로크꺼내기는 낡은 캐쉬내용들을 겹쳐쓰기하므로 적은 수의 블로크들은 블로크꺼내기 후에 즉시로 자료들이 재쓰기되는 결과를 나타낸다.
- 블로크가 더 커지면 블로크내의 다른 단어들은 요구되는 단어로부터 더 멀리 있게 되며 따라서 가까운 시간내에 참조될 가능성이 적게 된다.

블로크크기와 명중률사이의 관계는 개별적프로그람들의 국부성의 특성들에 의존하고 또한 결정적인 최량값을 찾을수 없으므로 복잡하다.

2 개로부터 8 개 단어까지 크기에 대하여서는 적당하게 최량값이 있을수 있다 [SMIT87a, PRZY88, PRZY90, HAND98].

캐쉬의 수

캐쉬들이 원리적으로 소개될 때 표준체계는 단일캐쉬를 가진다. 최근에 다중캐쉬의 리용이 일반적이다. 이 설계지표의 두가지 형태로는 여러준위 캐쉬방식과 동일준위분할 캐쉬리용방식이 있다.

론리밀도가 증가함에 따라 처리장치와 같은 소편에 캐쉬를 설치할수 있는 가능성이 생기게 된다. 외부모선을 통해서 도달할수 있는 캐쉬에 비하여 같은 소편우에 있는 캐쉬는 처리기의 외부모선활성을 감소시키므로 집행속도가 빨라 지게 하고 체계의 전체적인 성능을 높인다. 요구하는 명령이나 자료를 내부캐쉬에서 찾을 때 모선호출은 없어 진다. 처리장치안의 자료통로는 외부모선길이보다 비교적 짧으므로 내부캐쉬를 기다림상태가 없는 모선주기보다 현저하게 빨리 완성된다. 이 기간동안 모선은 다른 전송을 제공하기 위한 자유상태에 있게 된다. 내부캐쉬의 출현은 소편밖의 외부캐쉬가 여전히 있어야 하는가하는 질문을 가지게 한다. 대답은 있어야 한다는것이며 대부분 현대설계는 내부캐쉬와 외부캐쉬를 다 포함한다. 결과 이 구성을 2 준위캐쉬라고 하고 있다. 여기서 내부캐쉬는 L1(1 준위), 외부캐쉬는 L2(2 준위)이라고 한다. L2 준위캐쉬를 포함하게 되는 리유는 다음과 같다. L2 캐쉬가 없다면 처리장치는 L1 준위에서는 할수 없는 주기억기에 대한 호출요구를 가지게 된다. 이때 처리장치는 모선을 통하여 DRAM 혹은 ROM 기억기를 호출하여야 한다. 그러면 속도가 뜬 모선과 기억기호출속도로 인하여 낮은 성능을 초래하게 된다. 다른 한편 L2 SRAM 캐쉬가 리용된다면 자주 발생하는 캐쉬실패정보가 빨리 전송된다. SRAM 이 모선속도에 비하여 충분히 빠르다면 자료는 모선전송의 가장 빠른 형태인 기다림상태 없는 동작으로 호출될수 있다.

L2 캐쉬의 리용에 의한 가능한 보관은 L1 와 L2 캐쉬에서 명중률에 의존한다. 여러가지 연구들은 일반적으로 두번째 준위캐쉬의 리용이 성능을 개선한다는것을 보여 준다.

내부캐쉬가 처음으로 출현하였을 때 대부분 설계들은 자료와 명령을 함께 기억하는 단일캐쉬구성방식을 리용하였다. 최근에 이 캐쉬는 두개로 즉 명령을 기억하는 캐쉬와 자료를 기억하는 캐쉬로 가르는것이 일반적인것으로 되였다.

단일캐쉬방식은 다음의 여러가지 우점을 가진다.

- 주어 진 캐쉬크기에 대하여 단일캐쉬는 명령과 자료사이의 부하가 자동적으로 조절되므로 분할캐쉬방식보다 명중률이 높다. 즉 집행패턴이 재료꺼내기보다 더 많이 명령꺼내기를 진행하였다면 이때 캐쉬는 명령들로 가득차게 되며 집행패턴이 반대로 자료꺼내기를 많이 진행하였다면 반대현상이 나타난다.
- 하나의 캐쉬에 대해서만 설계하고 제작하면 된다.

이 우점을 무시하고 추세는 특별히 병렬명령집행과 예측된 선행명령들의 미리꺼내기를 강화한 Pentium II와 PowerPC와 같은 고속스칼라처리장치들에 쓰이는 분할캐쉬방향이다. 분할캐쉬설계의 기본우점은 이 방식이 명령처리장치와 집행장치사이에 캐쉬에 대한 경쟁을 없앤다는것이다. 이 방식은 명령의 관흐름조종에 기대를 거는 설계들에서 중요하다. 일반적으로 처리장치는 선행적으로 명령들을 미리꺼내여 완충기에 채워 넣으며 관흐름형식으로 처리한다. 이제 단일화된 명령/자료캐쉬를 가졌다고 가정하자. 집행장치가 자료를 꺼내고 기억하는 기억기호출을 진행할 때 호출요구는 단일캐쉬에 복종된다. 만일 동시에 명령미리꺼내기장치가 명령을 위하여 캐쉬에 읽기요구를 하였다면 이 요구는 현재 집행하고 있는 명령을 완성할수 있게 먼저 집행장치를 봉사하도록 봉쇄된다. 이 캐쉬경쟁은 명령관흐름조종방식의 효률 높은 리용에 간섭함으로써 성능을 떨어 지게 할수 있다. 분할캐쉬구조는 이 결함을 해결하였다.

제 4 절. Pentium II와 PowerPC캐쉬조직

1. Pentium II캐쉬의 조직

캐쉬조직의 발전을 인텔의 극소형처리장치의 발전에서 명백히 찾아 보자. 80386은 내부캐쉬를 포함하지 못하였다. 80486은 행의 크기가 16byte이고 4통로묶음련상방식으로 된 8Kbyte의 단일내부캐쉬를 포함하였다. Pentium Pro와 Pentium II도 2개의 L1내부캐쉬를 포함하였다. 처리장치의 초기변종들은 8Kbyte의 4통로묶음련상명령캐쉬와 8Kbyte의 2통로묶음련상자료캐쉬를 포함하였다. Pentium Pro와 Pentium II는 L1캐쉬들을 가지는 L2 캐쉬도 포함하였다. L2캐쉬는 256KB, 1MB사이의 크기를 가지는 4통로묶음련상방식이다.

그림 4-23에서는 3개의 캐쉬들의 배치가 명백한 Pentium II조직에 대하여 간단히 보여 주었다. 처리장치는 기본적으로 4개의 기본구성부분들로 구성되였다.

- **꺼내기/해신장치**: L1명령캐쉬로부터 순서대로 프로그람명령들을 꺼내며 마이크로연산들의 렬로 명령들을 해신하고 해신된 명령기다림렬에 해신결과를 기억한다.
- **해신된 명령기다림렬**: 집행가능한 해신된 명령들의 묶음을 보관한다.
- **선택/집행장치**: 자료의존성과 자원리용가능성에 종속되는 마이크로연산들의

집행을 순서화한다. 즉 마이크로조작들은 명령흐름으로부터 꺼내여 진 순서와 다른 순서로 집행순서를 작성한다. 시간이 허용되면 이 장치는 앞으로 요구되는 마이크로연산들의 투기적인 집행을 수행한다. 이 장치는 L1 자료캐쉬로부터 요구되는 자료를 꺼내고 등록기에 결과들을 림시적으로 기억하면서 마이크로연산들을 집행한다.

- **대피장치**: 림시적으로 언제 대피하였는가를 측정하여 등록기 혹은 L1 자료캐쉬에 영구적인 상태들로 투기적인 결과를 가지게 한다. 또한 이 장치는 결과들의 대피후에 해신된 명령대기렬로부터 명령들을 제거한다.

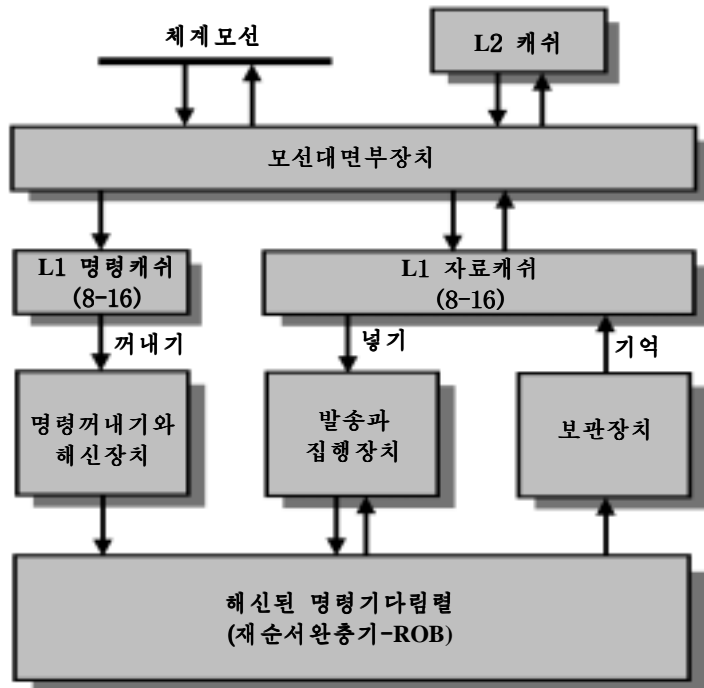



그림 4-23. Pentium II 블로크도

그림 4-24 에서는 L1 자료캐쉬의 기본구성요소들을 설명하였다. 캐쉬에서 자료는 두개의 행을 가진 128 개의 묶음들로 구성되였다. 이것은 두개의 4Kbyte 통로들로 론리적으로 구성되였다. 매개 행과 관련되는것은 표식표와 2 개의 상태비트들이다. 즉 이것들은 두개의 등록부로 구성되며 캐쉬의 매 행에 대하여 하나의 등록부항목을 가진다. 표식표는 대응하는 행에 기억된 자료의 주기억주소의 웃부분 24bit 이다. 캐쉬조종기는 LRU 방식의 치환알고리듬을 리용하며 따라서 한개의 LRU 비트는 두개의 행으로 된 매개묶음과 관련된다. 자료캐쉬는 비동시쓰기방식으로 되여 있다. 즉 자료는 캐쉬로부터 꺼내여 질 때 주기억에 쓰기되며 따라서 주기억기는 갱신된다. Pentium II 처리장치는 동시쓰기방식을 지원하도록 동적으로 구축될수 있다.

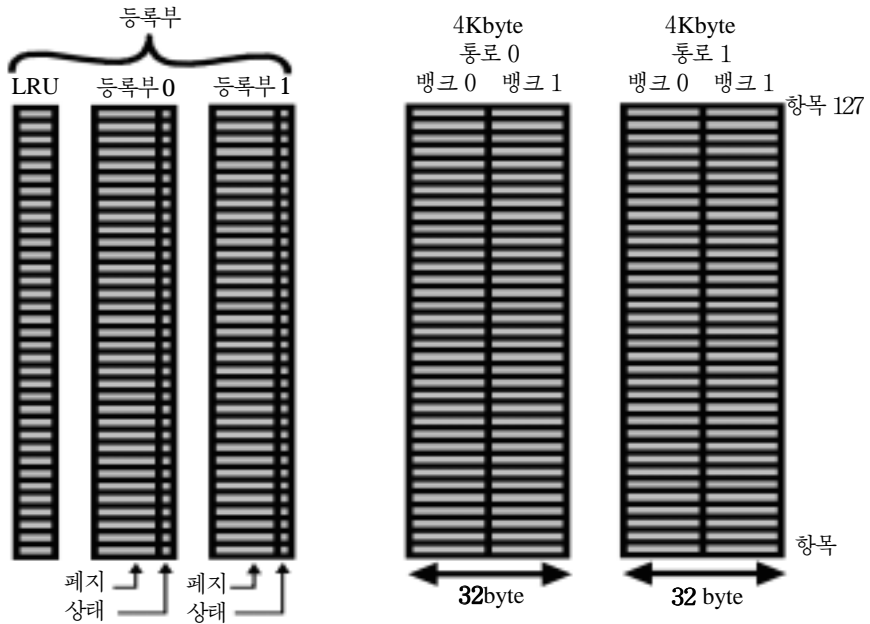



그림 4-24. Pentium II 자료캐쉬의 주소

자료캐쉬의 일치성

캐쉬의 일치성을 보장하기 위하여 자료캐쉬는 MESI 라는 규약을 지원하고 있다. MESI(modified/exclusive/shared/invalid)는 다중처리장치체계에서 캐쉬의 일치성요구들을 지원하기 위하여 설계되였다. 그러나 단일처리장치 Pentium II 조직에서도 효과적이다.

자료캐쉬는 표식자마다 2 개의 상태비트들을 가지며 따라서 매행은 4 개상태중의 하나로 될수 있다.

- **변경상태**: 캐쉬의 행이 변경되였을 때(주기억기와 다르다.) 이 캐쉬만 리용할수 있다.
- **독점상태**: 캐쉬의 행이 주기억기와 내용이 같으며 다른 캐쉬에는 존재하지 못한다.
- **공유상태**: 캐쉬의 행이 주기억기와 같으며 다른 캐쉬에 존재한다.
- **무효화상태**: 캐쉬의 행이 유효자료를 가지고 있지 못한다.

표 4-5 에 4 가지 상태들의 의미를 개괄하였다. MESI 규약은 제 16 장에서 상세히 론의한다.

표 4-5. MESI 캐쉬행상태

	M 변경	E 독점	S 공유	I 무효
캐쉬행은 유효인가?	유효	유효	유효	무효
기억기복사	자료넘침	유효	유효	—
다른 캐쉬에서 복사존재	불가능	불가능	가능	가능
행에 대한 쓰기	모선으로 못감	모선으로 못감	모선으로 가서 캐쉬갱신	모선으로 직접 감

캐쉬조종

내부캐쉬는 CD(cache disable)과 NW(not write-through)로 표기된 조종등록기들중의 하나에서 2 개의 비트들에 의하여 조종된다(표 4-6). 또한 캐쉬조종에 리용되는 2개의 Pentium II 명령이 있다. INVD는 내부캐쉬를 무효로 되게 하며 외부기억기를 무효로 하기 위한 신호를 보낸다. 외부캐쉬를 비동시쓰기로 하며 무효화한다.

표 4-6. Pentium II 캐쉬조작방식

조종비트		조작방식		
CD	NW	캐쉬	동시쓰기	무효화
0	0	가능	가능	가능
1	0	불가능	가능	가능
1	1	불가능	불가능	불가능

주의: CE = 0;NW = 1 은 무효화된 조합

2. PowerPC 캐쉬의 조직

PowerPC 캐쉬조직은 모든 극소형처리장치설계자들이 힘을 집중하고 있는 성능제고를 반영한 PowerPC 계렬의 전반적인 기본방식으로 개선되였다.

표 4-7. PowerPC 내부캐쉬

표준	크기	바이트/행	조직
PowerPC 601	132kbyte	32	8 통로묶음련상
PowerPC 601	28kbyte	32	2 통로묶음련상
PowerPC 601	216kbyte	32	4 통로묶음련상
PowerPC 601	232byte	64	8 통로묶음련상

표 4-7에는 이 개선과정을 보여 주었다. 기본형 601은 8통로묶음련상인 32Kbyte의 단일부호/자료캐쉬를 가지고 있다. 603은 더 강한 RISC설계를 실현하였지만 더 작은 캐쉬를 가지고 있다. 즉 두 통로묶음련상방식의 구성을 리용하였으며 개별적인 명령과 자료캐쉬로 분할된 16KB 크기로 되여 있다. 결과는 더 눅은 가격으로서 601 과 비슷한 성능을 가진다는것이다. 604와 620은 매개가 선행한 형에서의 캐쉬크기를 두배로 하였다. 현재 G3형은 620과 같은 L1캐쉬크기를 가진다.

그림 4-25에서는 두개의 캐쉬배치를 가진 PowerPC G3형의 블로크도를 보여 주

었다. 계렬의 다른 번호들의 구성도 비슷하다. 기본집행장치는 병렬집행가능한 두개의 옹근수 산수-론리연산장치와 류점수전용등록기들과 전용곱하기, 더하기, 나누기연산구성요소들로 되여 있는 류점수연산장치들로 되여 있다. 자료캐쉬는 load/store 장치를 통하여 옹근수와 류점수연산들에 연산수를 제공한다. 읽기전용명령캐쉬는 명령장치에 명령을 제공한다. 연산에 대해서는 제 13 장에서 론의한다.

L1 캐쉬는 8 통로 묶음련상방식이며 MESI 캐쉬일치성규약의 변종을 리용한다.

L2 캐쉬는 256K, 512K, 1MB 기억기로서 2 통로묶음련상캐쉬이다.

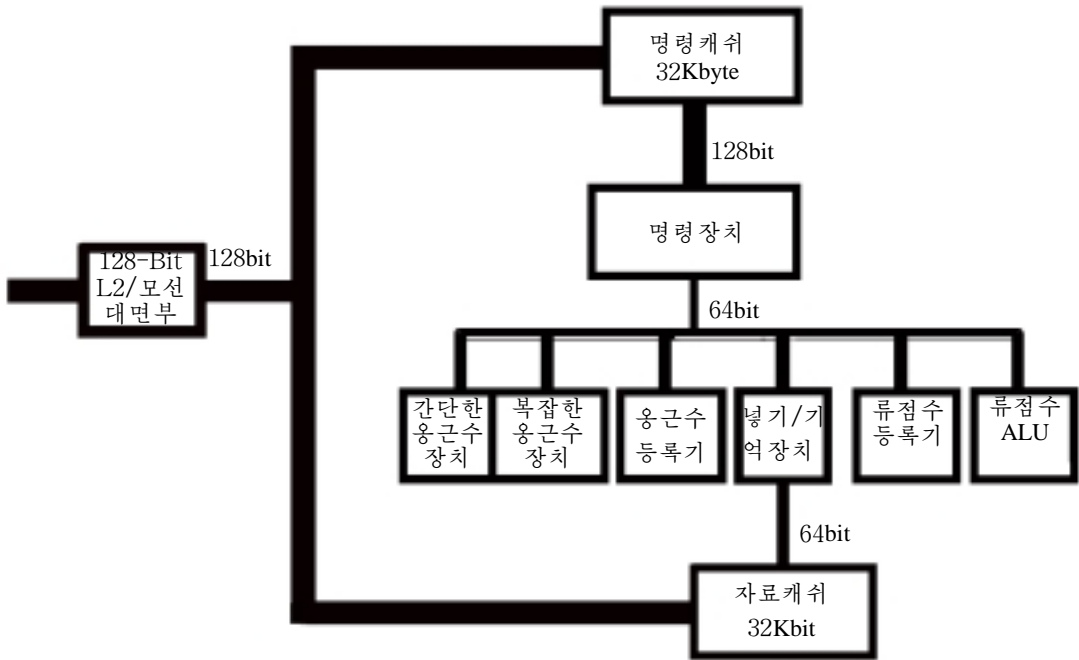



그림 4-25. Power PC G3 형 블로크도

제 5 절. 현대적인 DRAM 조직

제 2 장에서 론의한것처럼 고성능처리장치들을 리용할 때 가장 방해로 되는 체계의 하나는 기본내부기억기에 대한 결합방법이다. 이 결합방법은 전반적인 콤퓨터체계에서 가장 중요한 방법이다. 주기억기의 블로크를 구성하는 기초는 이미 1970 년대부터 DRAM 방식에서 기본적인 변화가 없으며 현재까지 20 년이상 리용하고 있는 DRAM 소편이다. 전통적인 DRAM 소편은 그의 내부방식과 처리장치의 기억기모선에 대한 결합방식에 제한되였다.

DRAM 주기억기의 성능문제를 해결하는 방법의 하나가 DRAM 주기억기와 처리장치사이에 하나이상의 고속 SRAM 캐쉬를 삽입하는것이라는것을 보았다. 그러나 SRAM

은 DRAM 보다 비용이 많이 들므로 이 점을 고려하지 않고 캐쉬용량을 확장하는것은 필요 없는 일이다.

몇년 지나서 기본 DRAM 방식을 보강한것들이 소개되였으며 그 일부는 지금 시장에서 판매되고 있다. 이 문제에서 하나의 표준적인 DRAM 으로서 출현하겠는가 아니면 여러가지가 다 존재하겠는가 하는것은 명백치 않다. 이 절에서는 이 새로운 DRAM 기술에 대하여 고찰한다.

1. 개선된 DRAM

대체로 가장 단순한 새로운 DRAM 방식은 램트론에 의하여 개발된 EDRAM 이다. EDRAM 은 일반적인 DRAM 소편에 적은 SRAM 캐쉬를 집적화하였다. 그림 4-26 은 EDRAM 의 4Mbit 변종을 소개하였다. SRAM 캐쉬는 2048bit 혹은 512 × 4bit 묶음으로 구성된 마지막으로 읽는 행의 전체 내용을 기억한다. 비교기는 가장 최근의 행주소선

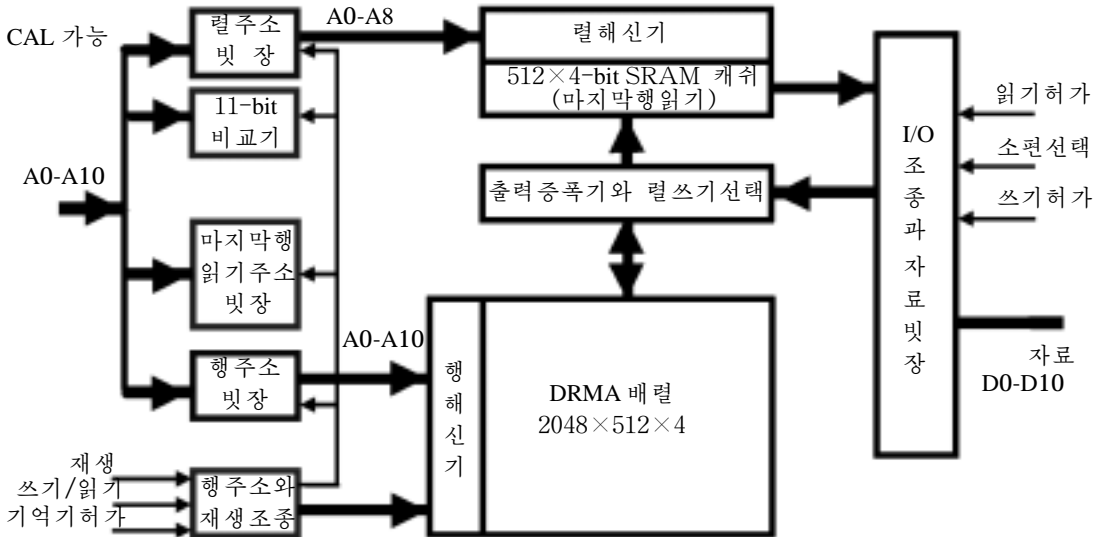



그림 4-26. 개선된 동적 RAM

택의 11bit 값을 기억한다. 다음호출이 같은 행이라면 그때 호출요구는 고속 SRAM 캐쉬에 대해서만 진행한다.

EDRAM 은 성능을 높이기 위한 여러가지 다른 특성들을 포함한다. 재생조작은 소편이 재생에 리용할수 없는 시간을 최소로 하면서 캐쉬읽기조작과 병렬로 진행할수 있다.

또한 행캐쉬로부터 출력포구까지의 읽기경로가 I/O 장치로부터 쓰기증폭기까지의 쓰기경로와 독립적이다. 이것은 쓰기조작의 완성과 병렬로 만족되게 캐쉬에 대한 련속적인 읽기호출을 가능하게 한다.

램트론에 의하여 작성된 연구론문들은 EDRAM 이 외부에 더 큰 SRAM 캐쉬를 가진 기본 DRAM 보다 더 좋거나 그와 비슷하게 수행한다는것을 지적한다.

2. 캐쉬 DRAM

미쯔비시에 의하여 개발된 캐쉬 DRAM(CDRAM)은 EDRAM과 비슷하다. DRAM은 EDRAM보다 더 큰 SRAM캐쉬(16~2KB)를 포함한다.

CDRAM에서의 SRAM은 2개의 통로로 리용할수 있다. 첫째로 여러개의 64bit행들로 구성된 실지 캐쉬로 리용될수 있다. 이것이 SRAM 캐쉬가 한개의 블로크 다시말하여 가장 최근에 호출된 행만으로 되여 있는 EDRAM과 대조적인것이다. CDRAM의 캐쉬방식은 보통 기억기에 대한 자유호출에 대하여 효과적이다.

CDRAM에서 SRAM은 자료블로크의 련속적인 호출을 지원하는 완충기로서 리용될수도 있다. 실례로 비트매프형 표시장치들을 재생하기 위하여 CDRAM은 DRAM으로부터 SRAM 완충기에로 자료를 미리 꺼내기할수 있다. 소편에 대한 련속적인 호출들은 단지 SRAM에 대한 호출로 된다.

3. 동기식 DRAM

DRAM성능을 개선하기 위한 완전히 다른 방법이 여러 회사들에서 협동개발된 동기식 DRAM(SDRAM)이다.

비동기식인 표준DRAM과 달리 SDRAM은 외부박자신호에 동기된 처리장치와 함께 자료를 교환하면 기다림상태가 없이 처리장치/기억기모선의 충분한 속도로 집행된다.

표준DRAM에서 처리장치는 주기억기의 개별적주소에서 자료묶음이 DRAM에 대하여 읽기 혹은 쓰기를 진행할수 있다는것을 가리키는 주소와 조종신호들을 가진다. 호출시간과 같은 일정한 지연후에 DRAM 은 자료에 대한 읽기 혹은 쓰기를 진행한다. 호출시간지연동안 DRAM은 행과 렬의 축전기들의 용량을 높게 하며 자료를 받아 들이고 출력완충기를 통하여 출력자료를 발송하는것과 같은 여러가지 내부기능들을 수행한다. 처리장치는 이 지연으로 약간 기다려야 하며 이것으로 체계성능은 떨어 진다.

동기식에서 DRAM 은 체계박자의 조종에 의하여 입출력자료를 전송한다. 처리장치나 다른 조종장치들은 DRAM에 의하여 빗장되는 명령과 주소정보를 전송한다. 이때 DRAM 은 여러개의 박자주기를 지나서 응답한다. 한편 조종장치는 SDRAM이 이 요구를 처리하고 있는 동안에 다른 과제를 수행할수 있다.

그림 4-27에서는 SDRAM의 내부론리를 보여 주었다. SDRAM은 첫 호출후에는 주소설정시간과 행과 렬로 선택된 자료를 먼저 충전하는 시간을 없애는 버스트방식을 채용하였다. 버스트방식에서 자료비트들의 련속적인 렬은 첫 비트가 호출된후에 빠르게 박자에 동기되여 출구될수 있다. 이 방식은 호출되는 모든 비트들이 련속적으로 되여 있으며 처음의 호출과 배렬이 같은 행에 있을 때 효과적이다.

더우기 SDRAM 은 소편내의 병렬화에 대하여 좋은 기회를 주는 2 중묶음내부방식을 가지고 있다.

방식등록기와 련관된 조종론리는 전통적인 DRAM 으로부터 SDRAM 들을 구별하는 또 하나의 기본특징이다. 이것들은 특별한 체계요구들을 만족하도록 SDRAM 을 요구대로

변경시킬수 있는 기구를 제공한다. 방식등록기는 모선에 동기적으로 주어 지는 자료의 분할된 단위들의 수인 버스트길이를 지적한다. 등록기는 또한 프로그람작성자가 읽기요구의 접수와 자료전송의 시작사이의 지연을 조절할수 있게 한다.

SDRAM 은 단어처리, 표계산, 다매체처리 등의 응용프로그람과 같은 련속적인 큰 자료블로크를 전송할 때 가장 좋은 성능을 발휘한다.

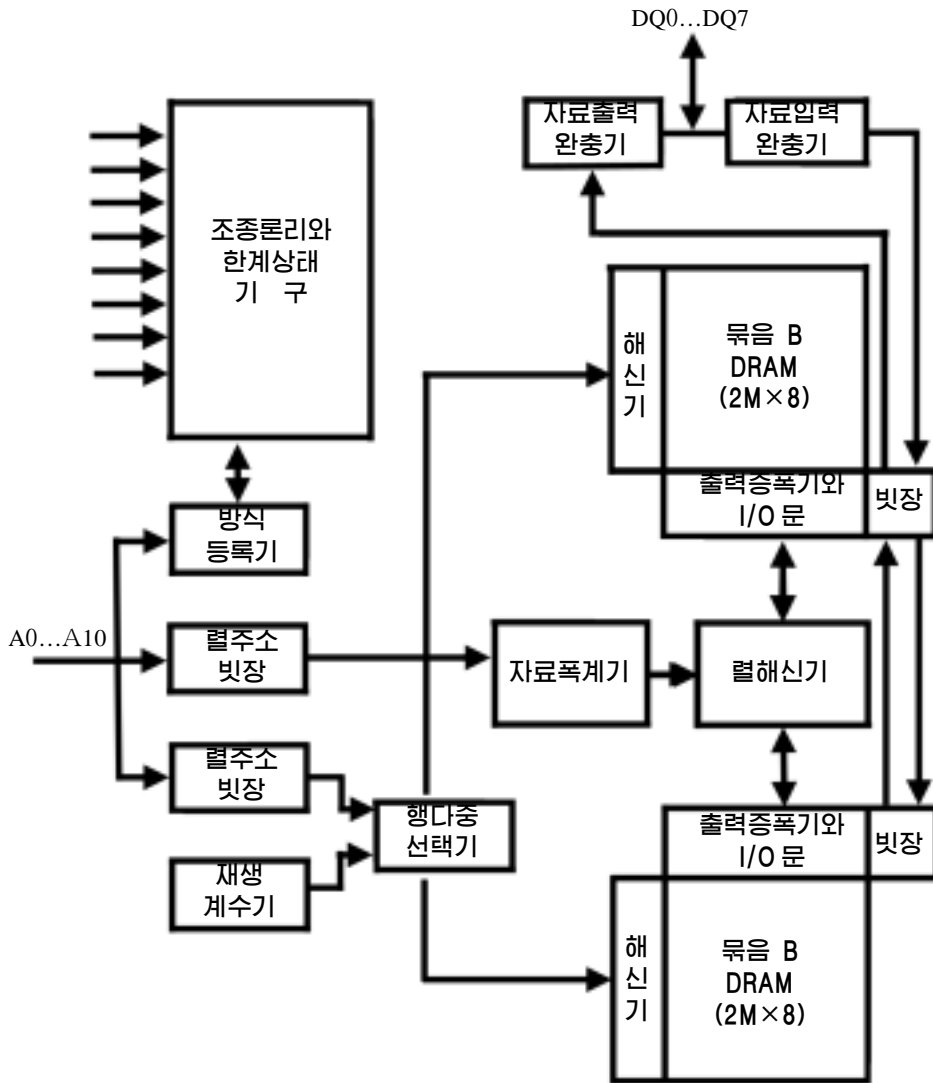



그림 4-27. 동기식 동적 RAM

4. 램바스 DRAM

램바스에 의하여 개발된 RDRAM 은 기억기대역폭문제에 대한 더 혁신적인 방법이다. RDRAM 소편은 모든 단자가 한쪽으로 되여 있는 수직팩키지형이다. 소편은 12Cm 길이보다 작은 28개의 선들로 처리장치와 자료를 교환한다. 모선은 320개

의 RDRAM 소편들에 대하여 주소화할수 있으며 500Mbps 정도의 속도를 가진다. 이것은 비동기 DRAM 들에서 33Mbps 에 대략적으로 비교한것이다.

전용 RDRAM 모선은 비동기적인 블로크지향통신규약을 리용하는 주소와 조종정보를 접수한다. 초기 480ns 호출시간후에 500Mbps 자료속도를 생성한다. 이 속도를 가능하게 하는것은 전압전류비, 박자화, 아주 정확한 신호들로 규정되는 모선그자체에 있다. 전통적인 DRAM 들에서 리용되는 명백한 RAS, CAS, R/W, CE 신호들에 의하여 조종되기보다 RDRAM 은 고속모선으로부터 기억기요구를 얻는다. 이 요구는 호출하려는 주소, 조작의 형태, 조작에 참가하는 바이트수를 포함한다.

5. 램련결

전통적인 DRAM 으로부터 가장 기본적인 변화는 SCI(Scalable Coherent Interface)라고 하는 IEEE 연구그루빠가 하나의 성과로서 개발한 램련결(RamLink)제품에서 볼수 있다. 램련결은 DRAM 소편들의 내부방식이 아니라 처리장치/기억기대면

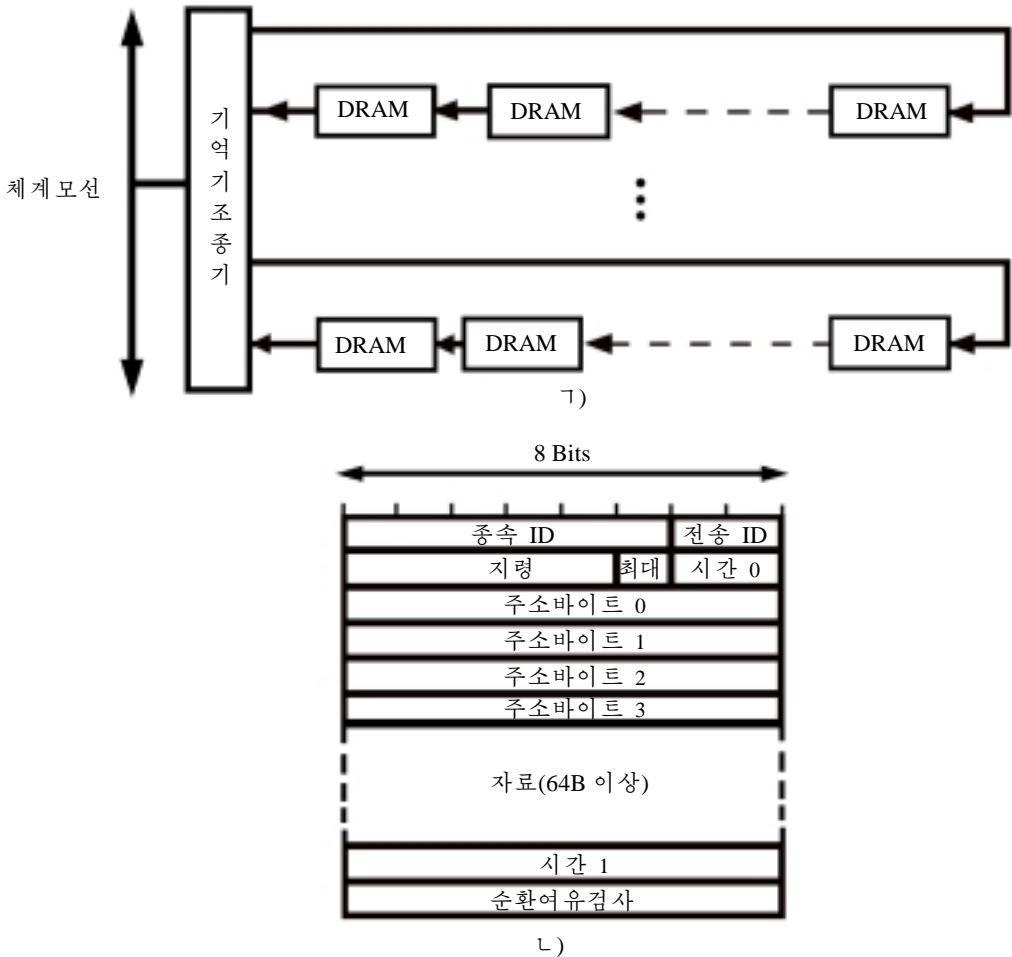



그림 4-28. 램련결

ㄱ- Ramlink 방식, ㄴ- 파케트형식

부에 집중하였다.

랩련결은 고리로 배렬된 점대점형식(point –to- point)의 기억기대면방식이다(그림 4-28 ㄱ). 고리의 통과는 DRAM 소편들(고리형망에서 마디들로서 작용하는)에 통보를 보내는 기억기조종기에 의하여 관리된다. 자료는 파케트형태로 교환된다(그림 4-28 ㄴ).

요구된 파케트는 기억기작용을 시동한다. 파케트들은 조종기에 의하여 보내여 지며 지령머리부, 주소검사합 그리고 쓰기인 경우 쓰기될 자료를 포함한다. 지령머리부는 형태, 크기 그리고 조종정보들로 구성되며 대응하는 종속기억기에 대하여 허용되는 규정된 응답시간 혹은 최대시간을 포함한다. 조종정보는 다음요구가 어느 렬주소에 있게 되겠는가하는것을 가리키는 하나의 비트를 포함하고 있다. 매 장치마다 이 네가지 작용이 동시적으로 진행될수 있다. 그러므로 모든 파케트들은 2bit 의 작용 ID 들을 가지고 요구와 정합해 보며 파케트에 애매한 점이 없이 응답한다.

성공적인 읽기에 대하여 DRAM 종속장치는 읽기자료를 포함하는 응답파케트를 보낸다. 실패한 요구에 대하여 종속장치는 동작을 완성하는데 얼마만한 시간이 요구되는가를 지적하는 상태파케트를 보낸다.

랩련결방법의 강력한 우점중의 하나는 적거나 많은 수의 DRAM 들을 지원하는 표준화된 방식을 제공하는것이며 내부 DRAM 조직을 지적하지 않아도 된다는것이다. 램련결의 고리형배렬은 많은 DRAM 들의 활용성을 동반하여 설계되였으며 기억기조종기에 대한 효률적인 결합을 제공한다.

램련결에서 최근 강화된것은 SDRAM 에 개발된 기술을 리용하는것이다. 현재 SLDRAM 이라고 한다.

참고문헌과 Web 싸이트

[PRIN91]은 SRAM, DRAM 그리고 플라쉬기억기들을 포함하는 반도체기억기기술에 대하여 포괄적으로 취급하고 있다. [SHAR97]은 검사와 믿음성의 문제점에 대한 보다 깊이 있는 고찰로 같은 반도체기억기에 대하여 제공한다. [PRIN96]은 현대적인 DRAM 과 SRAM 기본방식에 중점을 두었다.

오유수정부호에 대한 훌륭한 해석은 [MCEL85]에 있다. 더 깊게 학습하기 위하여 훌륭하고 충분하게 취급한것은 ADAM91]과 [BLAH83]이다. [SHAR97]은 오늘날 주기억기에서 리용된 부호들에 대한 훌륭한 개괄을 하고 있다.

캐쉬설계의 완전한 취급은 [HAND98]에서 찾아 볼수 있다. Pentium II 캐쉬조직의 자세한 서술은 [ANDE98]에서, PowerPC 캐쉬조직은 [MOTO97]과 [SHAN95]에서 찾아 볼수 있다. 더 가치 있는 론문은 [SMIT82]이다. 즉 론문은 캐쉬설계의 여러가지 요소들을 개괄하고 해석의 확장된 묶음에 대한 결과들을 주었다. 다중프로그람방식과 다중처리장치방식과 관련된 여러가지 캐쉬설계지표들의 자세한 설명들은 [AGAR89]에 주었다. [HIGB90]은 여러가지 캐쉬파라메터의 기능으로서 캐쉬성능을 평가하는데 리용하는 단순한 공식들을 주었다.

ADAM91 Adamek, J. *Foundations of coding*.New York: Wiley,1991
AGAR89 Agarwal, *Analysis of Cache Performance for Operating Systems and Multi-programming.* Boston: Kluwer Academic Publishers, 1989.
ANDE98 Adnerson, D., and Shanley, T.*Pentium Pro and Pentium II System Architecture*. Reading, MA: Addison-Wesley,1998.
BLAH83 Blahut, R.*Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.
HAND98 Handy, J. The *Cache Memory* book.san Diego: Academic Press, 1993.
HIGB90 Higbie, L. "Quick and Easy Cache Performance Analysis. "*Computer Archi-
tecture News*, June 1990.
MCEL85 McEiece, R. "The Reliability of Computer Memories. "*Scientific American*, January 1985.
MOTO97 Mororola, Inc. *PowerPC Microprocessor Family: The Programming Envi ronments for 32-bit Microprocessors*. Denver, CO, 1997
PRIN91 Prince, B. *Semiconductor Memories*. New York: Wiley, 1991
PRIN96 Prince, B. *High Performance Menories: New Architecture DRAMs and SRAMs, Evolution and Function* .New York: Wiley, 1996
SHAN95 Shanley. T. *PowerPC: System Architecture. Randing,* MA: Addison-Wesley, 1995.
SHAR97 Sharma, A. *Semiconductor memories:Technology,Testing,and Reliability.* New York: IEEE Press, 1997.
SMIT82 Smith, A."Cache Memories."*ACM Computing Surveys*, September 1992.

Web 싸이트

- **The RAM Guide**: 이 Web 싸이트는 RAM 기술과 그와 관련된 많은 쓸모 있는 기술들의 총적개괄을 제공한다.

련습문제

1. 왜 RAM 들이 전통적으로 소편당 1bit 로 구성되였으며 반대로 ROM 들은 소편당 여러개의 비트로 구성되였는가 하는 원인을 말해 보시오.
2. 1μs에 64번의 재생주기를 가지게 되는 동적 RAM을 고찰하자. 매번 재생조작 150ns 를 요구한다. 기억주기는 250ns 이다. 기억기 총 조작시간의 몇%가 재생에 리용되는가?
3. 64 × 1bit 크기의 SRAM 소편을 리용하여 총 용량이 8192bit 인 16 bit 기억기를 설계하시오. 기억기기판에서 소편의 배렬구성을 주고 제일 마지막공간에

배치하고 있는 기억기에 대하여 필요한 모든 입출력신호들을 표기하시오. 설계는 바이트와 16 bit 단어호출이 가능하게 하시오.

4. 기억기에 기억된 8 bit 자료단어가 11000010 이라고 하자. 하밍알고리듬을 리용하여 어떤 검사비트들이 기억기에 자료단어와 함께 기억되는가를 검사하시오. 또한 몇개의 답을 얻을수 있는가 찾아 보시오.
5. 8 bit 단어 00111001 에 대하여 검사비트들은 0111 로 기억된다. 단어가 기억기로부터 읽어 질 때 검사비트들은 1101 로 계산된다. 기억기로부터 읽어 진 자료단어는 어떻게 되는가?
6. 하밍오유수정부호가 1024 bit 자료단어에서 한개의 비트오유들을 검사하는데 리용된다면 얼마나 많은 검사비트들이 필요한가?
7. 16 bit 자료단어에 대한 SEC 부호를 개발하시오. 자료단어 0101000000111001 에 대한 부호를 생성하시오. 부호가 자료비트 4 에서의 오유를 정확히 확정하는 경우를 고찰하시오.
8. 묶음련상캐쉬는 묶음당 4 개의 행으로 된 64 개의 행들로 이루어 졌다. 주기억기는 매개가 128 단어들로 된 4K 개의 블로크를 포함한다. 주기억기주소의 형식을 고찰하시오.
9. 16 진수기억주소 111111, 666666, BBBBBB 에 대하여 16 진수형식으로 다음의 정보들을 고찰하시오.

   ㄱ. 그림 4-18 의 형식을 리용하여 직접배치방식에서 표식표, 행, 단어값
   ㄴ. 그림 4-20 의 형식을 리용하여 완전련상배치방식에서 표식표와 단어값
   ㄷ. 그림 4-22 의 형식을 리용하여 두 통로묶음련상배치방식에서 표식표, 묶음, 단어값

10. 4 통로묶음련상배치방식인 16KB 의 내부캐쉬를 가진 32bit 극소형처리장치를 고찰하자. 캐쉬가 4 개의 32bit 단어들로 한개 행을 구성한다고 가정하자. 이 캐쉬에 대한 블로크도를 그리고 구성을 고찰하고 캐쉬명중률을 측정하는데 서로 다른 마당들이 얼마나 리용되는가. 기억주소 ABCDE8F8의 단어가 캐쉬에 어디에 배치되는가? 참고[ALEX93]
11. 외부캐쉬에 대하여 4통로 묶음련상방식, 2개의 16bit 단어의 크기로 된 행, 주기억으로부터 총 4K 의 32bit 단어들을 조종할수 있다는 조건이 주어 졌고 24bit 주소를 가진 16bit 처리장치를 리용할 때 캐쉬구조와 알맞는 정보들을 설계하고 처리장치의 주소들을 어떻게 분할할수 있는가를 고찰하시오. 참고[ALEX93]
12. 인텔의 80486 은 단일방식의 내부캐쉬를 가지고 있다. 용량은 8KB이며 4통로 묶음련상구성이고 블로크의 길이는 4 개의 32bit 단어로 되여 있다. 캐쉬는 128 개의 묶음으로 되여 있다. 묶음마다 한개의 《행유효비트》와 LRU 비트들인 B0, B1, B2 비트들이 있다. 캐쉬실패때에 기억기가 버스트읽기방식으로된 모선에서 80486 은 주기억으로부터 16byte 행을 읽는다. 캐쉬의 간단한 그림을 그리고 주소의 서로 다른 마당들이 어떻게 분할되는가를 고찰하시오. 참고[ALEX93]

13. 바이트주소가능한 2^{16}byte 용량을 가진 주기억기가 8byte 의 블로크크기로 분할된다고 하자. 32 행으로 구성된 직접배치형캐쉬가 이 기억체계에 리용된다고 가정하자.

ㄱ. 16bit 기억주소는 꼬리표, 행번호, 바이트번호로 어떻게 분할되는가?

ㄴ. 어느 행에 다음주소를 대응하는 바이트들이 기억되는가?

0001 0001 0001 1011
1100 0011 0011 0100
1101 0000 0001 1101
1010 1010 1010 1010

ㄷ. 주소 0001 1010 0001 1010 의 바이트가 캐쉬에 기억되였다고 하자. 이 주소를 따라 기억된 다른 바이트들의 주소는 무엇인가.

ㄹ. 기억기의 바이트가 최대로 얼마나 많이 캐쉬에 기억될수 있는가.

ㅁ. 꼬리표는 왜 캐쉬에 기억되는가?

14. 인텔 486 의 재배치알고리듬은 가짜 LRU 방식으로 알려 졌다. L0, L1, L2, L3 으로 표기된 4 개의 행으로 된 128 개의 묶음의 매개와 관련되는것은 B0, B1, B2 비트들이다. 재배치알고리듬은 다음과 같이 동작한다. 어떤 행을 재배치하여야 할때 캐쉬는 처음으로 가장 최근에 리용된것이 L0과 L1로부터 혹은 L2과 L3 으로부터 주어 지는가를 검사하여야 한다. 그다음 캐쉬는 블로크쌍중의 어느것이 가장 최근에 리용되였는가를 검사하고 재배치를 위한 블로크를 선택한다.

ㄱ. 비트 B0, B1, B2 가 어떻게 설정되며 재배치알고리듬에 어떻게 리용되는가를 규정하시오.

ㄴ. 80486 알고리듬이 진짜 LRU 알고리듬과 근사하다는것을 고찰하시오.

ㄷ. 진짜 LRU 알고리듬은 묶음당 6 개의 비트를 요구한다는것을 증명하시오.

15. 묶음련상캐쉬는 4 개의 16bit 단어로 된 블로크를 가지며 2 개 블로크로 된 크기의 묶음을 가진다. 캐쉬가 총 4048 단어를 관리할수 있다. 캐쉬에 완충가능한 주기억기크기는 64K×32bit 이다. 캐쉬구조를 설계하고 처리장치의 주소가 어떻게 분할되는가를 고찰하시오. 참고[ALEX93]

16. 4 통로 묶음련상캐쉬에서 LRU재배치알고리듬을 실현하기 위한 간단한 기술을 서술하시오.

17. 다음프로그람을 고찰하자.

```
for (i = 0; i < 20; i++)
    for (j = 0; j < 10; j++)
    a[i] = a[i] * j
```

ㄱ. 프로그람에서 공간적국부성의 한 실례를 드시오.
ㄴ. 프로그람에서 시간적국부선의 한 실례를 드시오.

18. N 개준위기억기의 계층을 위한 부록 4 의 식 4-1 과 식 4-2 를 설명하시오.
19. 콤퓨터체계는 32K 16bit 단어의 주기억기를 가지고 있다. 또한 한행이 64 단어이고 4 개 행을 가지는 묶음으로 분할된 4K 단어캐쉬를 가지고 있다. 캐쉬가 초기에 비였다고 가정하자. 처리장치는 주소 0, 1, 2, …, 4351 로부터 순서대로 단어들을 꺼낸다. 그다음 이 꺼내기를 순차적으로 9 번 더 반복한다. 캐쉬는 주기억보다 10 배 더 빠르다. 캐쉬의 리용으로부터 나타나는 성능개선을 평가하시오. 블로크재배치를 LRU 방법이라고 가정한다.
20. 다음과 같은 파라메터들을 가진 기억체계를 고찰하자.

T_C = 100ns　　C_C = 0.01 쎈트/bit

T_m = 1200ns　　C_m = 0.001 쎈트/bit

ㄱ. 1Mbyte 기억기의 가격은 얼마인가?
ㄴ. 캐쉬기억기기술을 리용한 1Mbyte 주기억기의 가격은 얼마인가?
ㄷ. 실제 호출시간이 캐쉬호출시간보다 10% 크다면 캐쉬명중률 H 는 얼마인가?

21. 콤퓨터는 캐쉬, 주기억기 그리고 가상기억으로 리용되는 디스크를 가지고 있다. 참조되는 단어가 캐쉬에 있다면 그 호출에 20ns 가 요구된다. 주기억기에만 있고 캐쉬에 없다면 그 단어를 캐쉬에 넣기하는데 60ns 요구되며 참조시간은 처음의 경우와 같다. 만일 단어가 주기억기에 없다면 디스크로부터 단어를 꺼내는데 12ms 가 60ns 에 의하여 캐쉬 h 에 복사되며 첫 경우와 같이 참조가 진행된다. 캐쉬의 명중률은 0.9 이며 주기억기의 명중률은 0.6 이다. 이 체계에서 참조되는 단어를 호출하는데 요구되는 평균시간은 몇 ns 인가?

부록 4. 2 준위기억기의 성능특성

이 장에서 기억참조는 2 준위내부기억기로 구성되면서 주기억기와 처리장치사이에 완충기로서 작용하는 캐쉬에서 이루어 진다. 2 준위방식은 부록에서 소개하는 국부성의 성질을 실현함으로써 1 준위기억기에 비하여 우월한 성능을 가진다.

주기억기캐쉬기구는 콤퓨터기본방식의 한 부분이며 장치적으로 실현되고 조작체계에 의해서는 볼수 없다. 국부성을 실현한 2 준위기억방법의 두가지 다른 실례들이 있으며 그것은 마지막부분으로서 조작체계에 의하여 실현된다. 즉 가상기억기와 디스크캐쉬를 말한다(표 4-8). 가상기억기는 제 7 장에서 주었다. 디스크캐쉬는 이 책의 범위를 넘어 서므로 [STAL98]에서 찾아 보시오. 이 부록에서는 모두 세가지 방법으로 공

통인 2준위기억기들의 일부 특성을 고찰한다.

표 4-8. 2준위기억기의 특성

	주기억기캐쉬	가상기억기(페지방식)	디스크캐쉬
일반적인 호출속도	5/	1000/1	1000/1
기억기관리체계	특수한 장치로 실현	장치와 체계프로그람의 결합	체계프로그람
일반적인 블로크 크기	4～128byte	64～4096byte	64～4096byte
2차준위에 대한 처리 장치의 호출	직접호출	간접호출	간접호출

1. 국부성

2 준위기억기의 성능개선을 위한 기초는 **참조의 국부성**이라고 하는 원리이다 [DENN68]. 이 원리는 기억기참조가 클라스터에 귀착된다는것을 반영하고 있다. 오랜시간 지나면 리용에서 클라스터들이 교체되지만 짧은시간 동안에는 처리장치가 기억기참조로서 고정된 클라스에서 기본적으로 작업한다.

국부성의 원리는 직관적으로 알수 있다. 다음의 추리방향을 고찰하자.

- 모든 프로그람명령중에 적은 부분을 차지하는 분기와 호출명령을 제외하면 프로그람집행은 순차적이다. 결국 대부분의 경우에 꺼내여 지는 다음명령은 현 시점에서 마지막으로 꺼낸 명령에 직접 이어 진다.
- 호출명령에 의하여 호출되는 보조프로그람들이 끊어 짐이 없이 매우 길어 지게 되는 경우는 거의 없다. 오히려 프로그람은 틀의 심도가 어느 정도 좁은 창문(부분)으로 되도록 구성한다. 따라서 짧은 시간동안에는 명령들에 대한 참조가 작은 틀들에 대하여 국부적으로 진행될 경향성이 많다.
- 대부분 반복적인 구성들은 많은 회수로 반복되는 상대적으로 적은 수의 명령들로 이루어 진다. 그러므로 반복되는 지속시간에 대하여 계산을 프로그람의 작은 린접적인 부분에 극한된다.
- 많은 프로그람들에서 대부분의 계산은 배렬 혹은 레코드들의 렬과 같은 처리자료구조를 가진다. 많은 경우에 이 자료구조에 대한 련속적인 참조는 엄밀하게 국부적인 항들에 대한것이다.

이 추리방향은 많은 연구들에 의하여 확증되였다. 첫번째 추리방법에 대하여 고수준언어프로그람들의 동작상태들을 분석하였다. 표 4-9 는 다음과 같은 연구들로부터 집행중에 여러가지 명령형태들의 출현을 측정한 기본결과들을 포함하였다. 크느츠[KNUT7]에 의하여 완성된 프로그람언어구성형식에 대한 초기연구는 대학생실험으로서 리용된 FORTRAN 프로그람의 묶음을 조사하였다.

표 4-9. 고수준언어연산의 상대적인 동적주기

학 습	[HUCK83]	[KNUT71]	[PATT82]		[TANE78]
언 어	Pascal	FORTRAN	Pascal	C	SAL
용 도	과학	대학생	System	System	System
Assign	74	67	45	38	42
Loop	4	3	5	3	4
Call	1	3	15	12	12
IF	20	11	29	43	36
GOTO	2	9		3	
Other		7	6	1	6

탠넌바움[TANE78]은 조작체계에서 리용되고 구조체프로그람작성법(SAC)을 지원하는 언어로 작성된 300 개 틀들로부터 얻어 진 측정들을 공개하였다. 페터순과 씨쿼인[PATT82a]는 콤파일러들과 편집기, CAD, 정렬프로그람(sorting)과 파일비교프로그람들로 부터 주어 진 측정들의 묶음을 분석하였다. 프로그람작성언어인 C 와 파스칼이 연구되였다. 후크는 [HUCK83] 고속푸리에변환과 미분방정식들의 통합체계들을 포함하는 혼합형일반용과학계산을 위한것들이 4 개의 프로그람들을 분석하였다. 분기와 호출명령들이 프로그람이 집행되는 동안 집행되는 명령들의 극히 일부만이 존재하는 언어들과 응용프로그람들의 경우에는 좋은 결과를 얻는다. 따라서 이 연구들은 주장 1 을 확정하였다.

주장 2 에 대해서는 [PATT85a]에 보고된 연구들이 이 주장의 증명을 제공한다. 이것은 Call-return 구성을 보여 주는 그림 4-29 에서 설명하였다. 매 호출은 오른쪽으

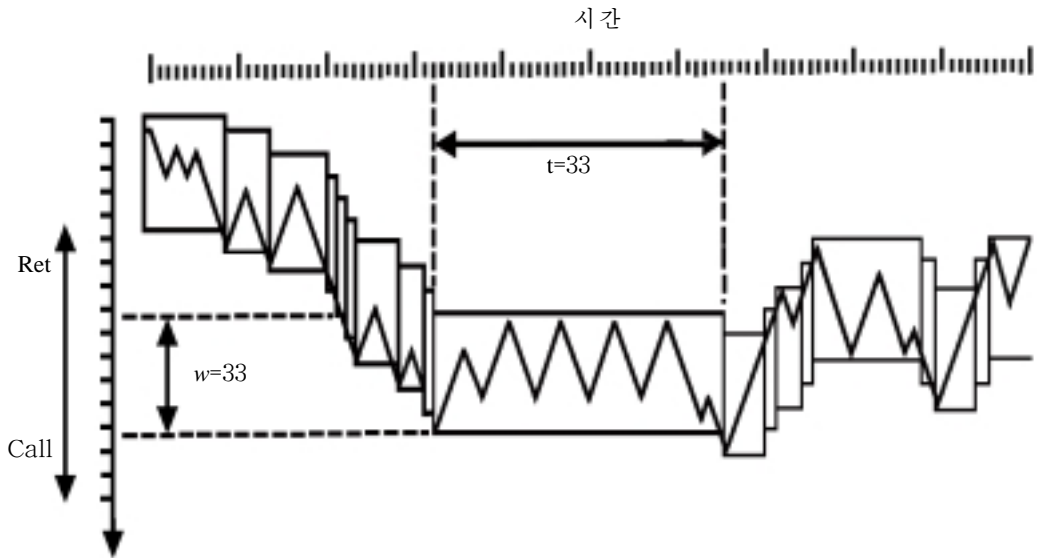



그림 4-29. 프로그람의 Call/Rreturn 동작과정

로 이동하면서 아래로 향한 선들로, 매개 되돌이는 오른쪽으로 이동하면서 우로 향한 선들로 표시되였다. 그림에서 길이가 5 인 창문에 대하여 보여 주었다. 다른 한편으로 오직 6 개의 봉우리들로 된 련속적인 호출과 되돌이들은 창문내에서 발생한다. 보는바와 같이 집행하고 있는 프로그람은 오랜 시간 고정된 창문에서 유지된다. C 와 파스칼 프로그람들에 대한 같은 분석자들의 연구는 길이가 8 인 창문을 호출이나 되돌이들의 1%이하로만 바뀌 지는것을 요구한다[TAMI83].

국부성참조의 원리는 최근 연구들에서 계속 유효하게 되였다. 실례로 그림 4-30 에 하나의 기지에서 Web 페지호출패턴들에 대한 결과를 주었다. 공간적국부성과 시간적국부성은 서술에서 차이가 있다. **공간적국부성**은 묶음으로 된 여러개의 기억주소들에서 집행의 경향성이 있다고 말하는것이다. 이것은 자료들의 표를 처리할 때와 같이 순차적으로 명령들을 호출하는 처리장치의 경향성에 따른다. **시간적국부성**은 최근에 리용된 기억주소들을 호출하는 처리장치에서의 경향성이라고 말하는것이다. 실례로 반복순환이 집행될 때 처리장치는 반복적으로 같은 명령들의 모임을 집행한다.

2. 2 준위기억기의 조작

루적된 문서수작성과정

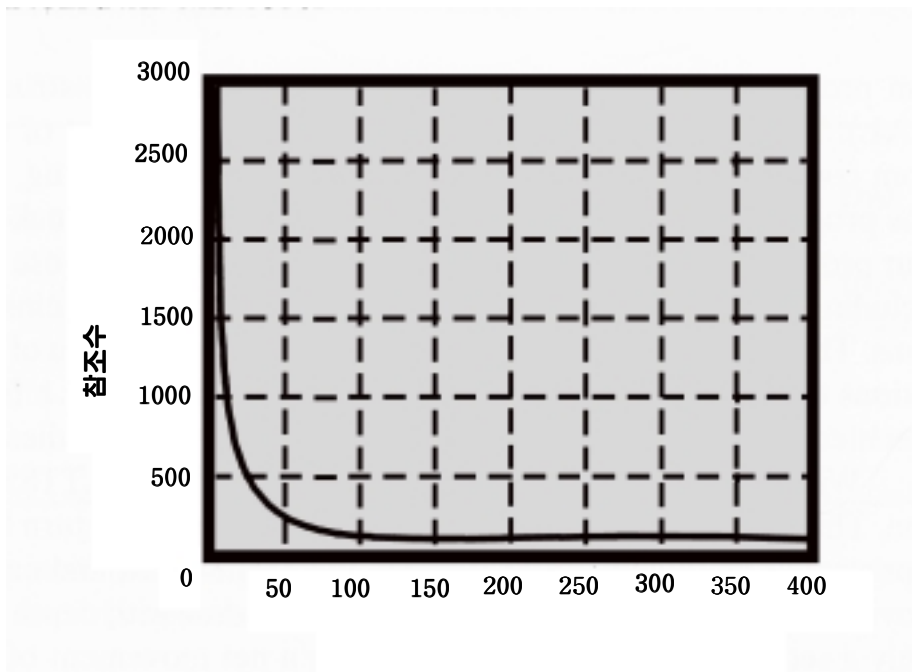



그림 4-30. Web 페지에 대한 참조의 국부성

국부성의 성질을 2 준위기억기형식에서 얻을수 있다. 웃준위기억기(M1)는 아래준위기억기(M2)보다 기억용량이 작고 속도가 빠르며 더 비용이 많이 든다. M1 는 용량이 큰 M2의 내용들의 일부분을 림시기억하는데 리용된다. 기억기참조가 이루어 질 때 출발은 M12 의 항을 호출하도록 한다. 만일 성공이라면 빠른 호출이 이루어 진다. 만일 실패라면 해당 기억기블로크가 M2 로부터 M1 로 복사되며 M1 에 배치된 조건에서

호출을 진행한다. 국부성으로 인하여 일단 블로크가 M1 로 전송되면 블로크내의 주소들은 여러번 호출될수 있으며 전반적인 봉사가 빠르게 진행된다.

항목을 호출하는 평균시간을 식으로 표시하기 위하여 2 개 준위의 기억기들의 속도뿐아니라 주어 진 참조가 M1 에서 진행될 확률까지로 고찰하여야 한다,

$$T_s = H \times T_1 + (1 - H) \times (T_1 + T_2) = \tag{4-1}$$
$$= T_1 + (1 - H) \times T_2$$

여기서 T_s - 평균(체계)호출시간,

T_1 - 평균(체계)호출시간,

T_2 - M1(캐쉬, 디스크캐쉬 등)의 호출시간,

H - 명중률(호출되는 부분이 M1 에 있는 경우)

그림 4-2 에서는 명중률의 함수로서 평균호출시간을 보여 주었다. 보는바와 같이 명중률이 높으면 평균총호출시간이 M2 보다 M1 의것에 가까워 진다.

3. 성능

2 준위기억장치의 평가를 위하여 관련되는 일부 파라메터들을 고찰하자.

처음으로 가격을 고찰하자.

$$C_s = \frac{C_1 S_1 + C_2 S_2}{S_1 + S_2} \tag{4-2}$$

여기서 C_s - 결합된 2 준위기억기에서의 비트당 평균가격,

C_1 - 웃준위기억기 M1 에서 비트당 평균가격,

C_2 - 아래준위기억기 M2 에서 비트당 평균가격,

S_1 - M1 의 크기,

S_2 - M2 의 크기

기억기체계에서 $C_s \approx C_1$로 되여야 한다. $C_1 \gg C_2$로 주어 지고 $S_1 \ll S_2$일것을 요구한다. 그림 4-31 에서는 이 관계를 보여 주었다.

다음으로 호출시간을 고찰하자. 충분한 성능개선을 제공하는 2 준위기억기에 대하여 T_1와 근사하게 되는 T_s($T_s \approx T_1$)를 가질것을 요구한다. T_1가 T_2보다 대단히 작게 ($T_1 \ll T_2$) 주어 지고 1 에 가까운 명중률이 요구된다.

이렇게 명중률과 그에 의한 성능이 높으며 가격을 눅게 하는 작은 M1 로 되여야 한다. 이 론리적인 크기에 대한 두가지 요구를 만족하는 M1 의 크기가 있는가? 우리는

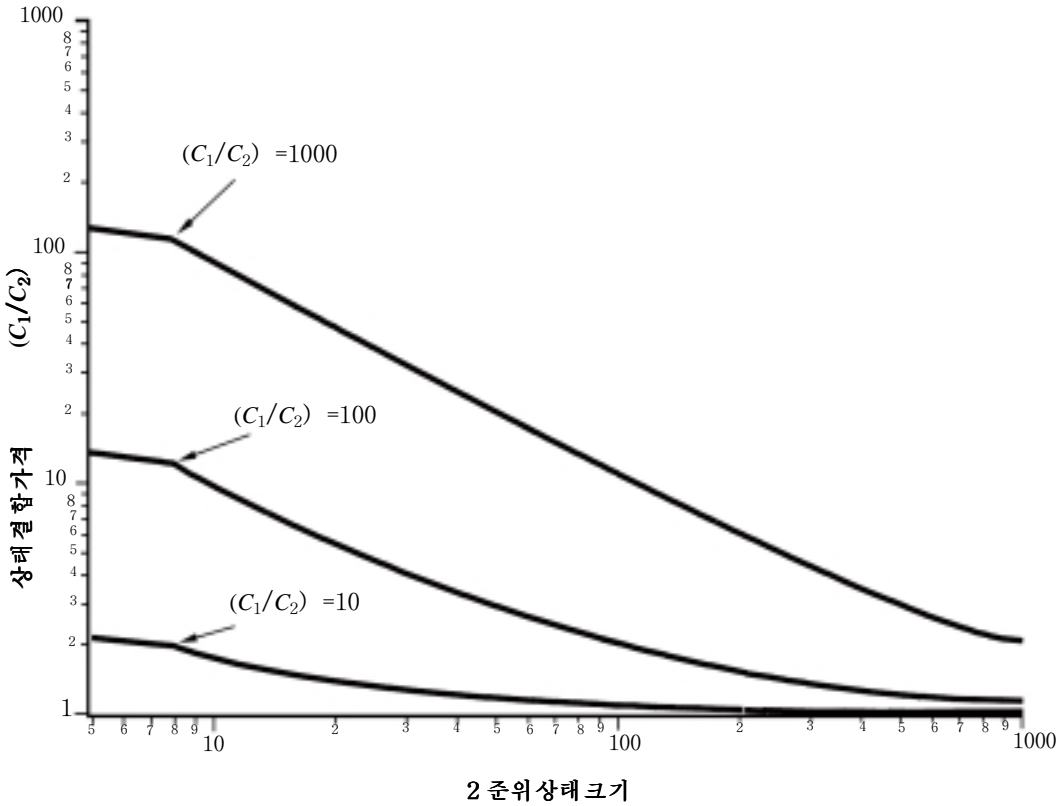



그림 4-31. 2 준위기억기에서 기억기크기와 평균기억기가격의 관계

다음물음들로 이 질문에 대답할수 있다.

- 성능요구를 만족하기 위하여 얼마만한 명중률이 요구되는가?
- 요구되는 명중률은 M1 의 얼마만한 크기의 M1 를 담보할수 있는가?
- 이 크기는 가격의 요구를 만족시키는가?

이 대답을 얻기 위하여 호출효률이라고 하는 T_1/T_S 량을 고찰하자. 평균호출시간(T_S)이 M1 호출시간(T_1)에로 어떻게 다가가는가 하는 측정을 하자.
식 4-1 로부터

$$\frac{T_1}{T_2} = \frac{1}{1+(1+H)\dfrac{T_2}{T_1}} \qquad (4\text{-}3)$$

이 주어 진다.

그림 4-32 에서 파라메터로서 T_2/T_1 량과 명중률 H 의 함수로서 T_1/T_S 를 그리였다.

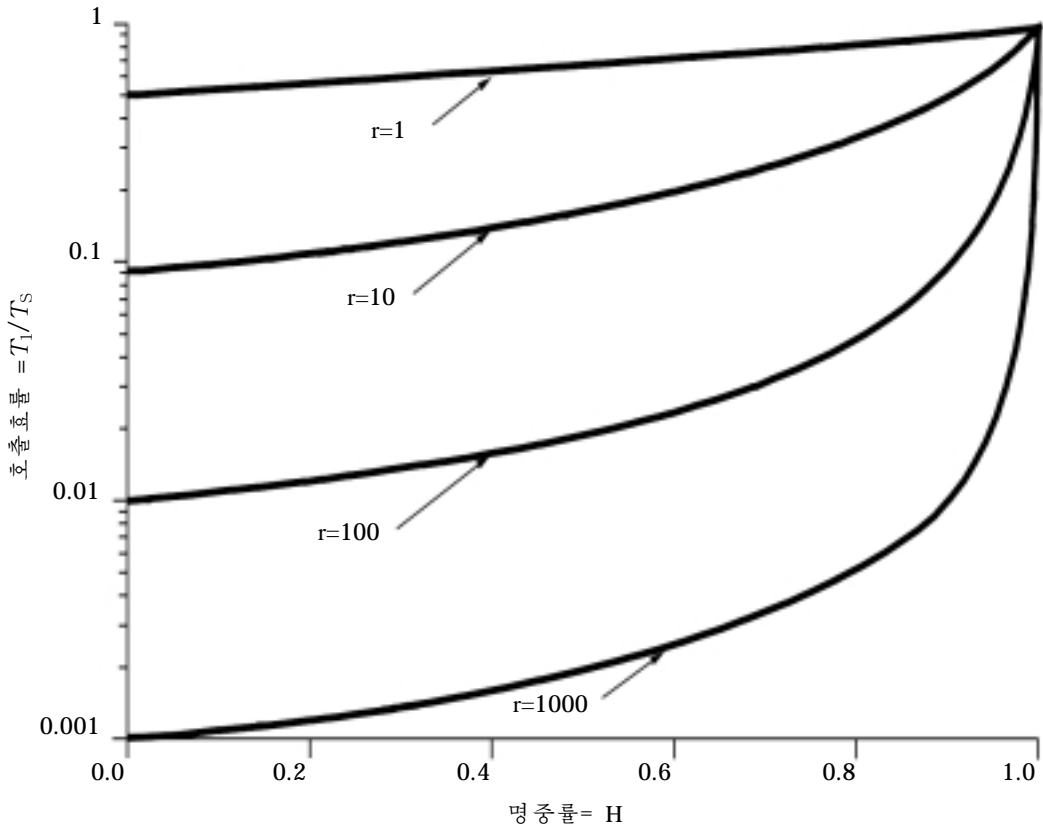



그림 4-32. 명중률함수로서의 호출효률 ($r=T_2/T_1$)

일반적으로 캐쉬호출시간은 주기억기호출시간보다 대략 5 ～ 10 배(T_2/T_1 = 5~10 배) 빠르며 주기억기호출시간은 디스크호출시간보다 약 1000 배정도 빠르다(T_2/T_1 = 1000). 따라서 0.8～0.9 범위의 명중률이 성능요구를 만족시키는데 필요로 된다. 더 정확하게 상대적인 기억기크기에 대한 질문을 제기할수 있다. $S_1 \ll S_2$ 에 대하여 0.8 의 명중률이나 그이상 더 적당한것이 있을수 있는가?

이것은 집행하고 있는 프로그람의 내용과 2 준위기억기의 설계세부를 포함하는 여러인자들에 의존한다. 물론 기본적으로 명백한것은 국부성의 정수이다. 그림 4-33 은 국부성이 명중률에 미치는 효률을 시사하였다. 명백히 M1 과 M2 가 크기가 같으면 명중률은 1 일것이다. M2 에서의 모든 항들은 항상 M1 에 기억된다. 이제 국부성이 없다고 가정하자. 즉 참조는 란수적으로 실현된다고 하자. 이 경우에 명중률은 기억기비률에 대하여 완전한 선형함수이다. 실례로 M1 과 M2 의 크기에 절반이라면 이때 명중률은 0.5 이다. 그러나 실제적으로 참조에는 어느 정도 국부성이 있다. 중간정도와 강한 국부성의 효률을 그림 4-33 에 주었다.

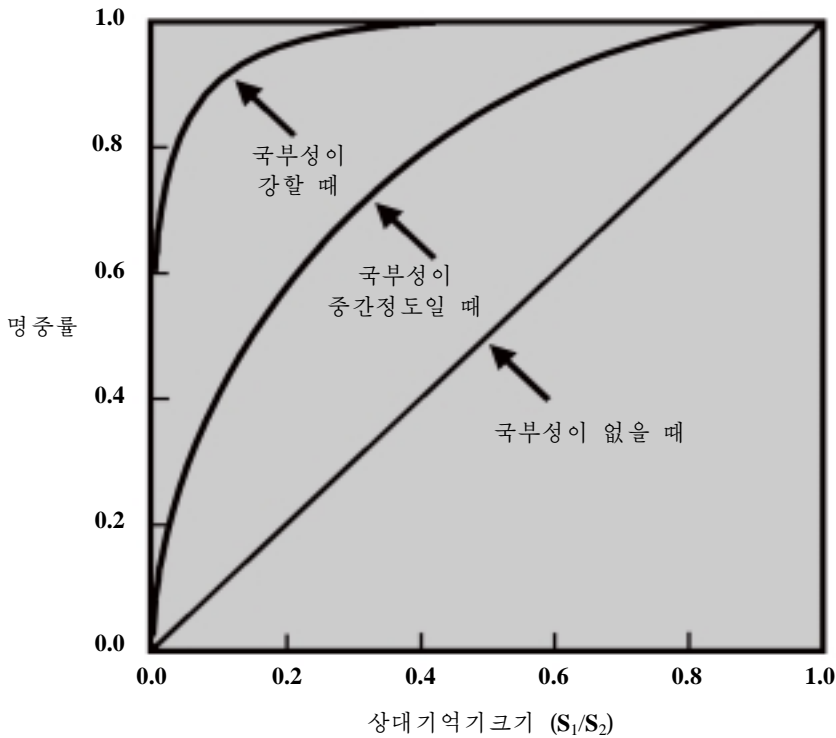



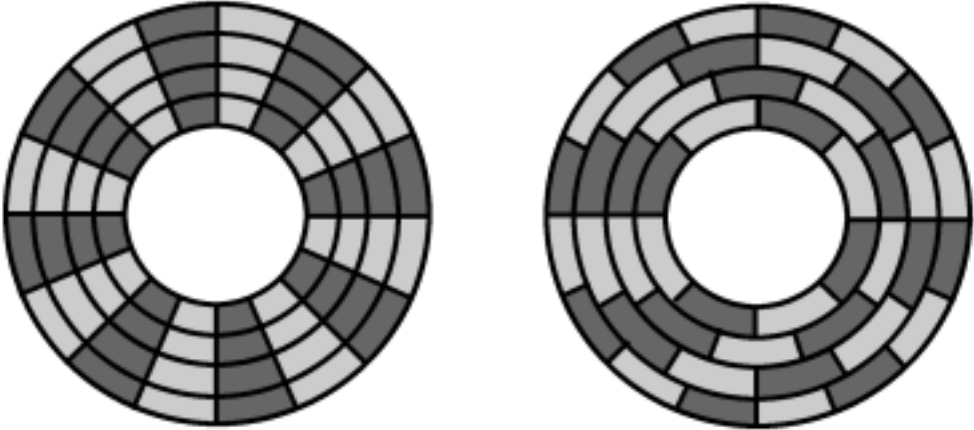
그림 4-33. 상대기억기크기에 대한 명중률

만일 국부성이 강하면 상대적으로 작은 크기의 높은 준위기억기라고 하여도 높은 명중률을 달성하는것이 가능하다. 실제로 여러 연구들은 오히려 주기억기크기에 관계없이 작은 크기의 캐쉬들이 약 0.75의 명중률을 가질수 있다는것을 보여 주었다. ([AGAR89a], [PRZY88], [STRE83], [SMIT82], …). 주기억기가 현재 수 MB 범위에있는 사실에 비추어 보면 1K로부터 128K 단어범위캐쉬들이 일반적으로 적당하다.

가상기억기와 디스크캐쉬를 고찰할 때 같은 현상을 확정하는 다시말하여 상대적으로 작은 M1가 국부성에 의하여 높은 명중률을 가진다는것을 확정하는 다른 연구들을 소개할수 있다.

이것은 이미 제기된 마지막질문을 준다. 즉 두 기억기의 상대적크기가 가격상 요구를 만족시키는가? 대답은 명백히 《예》이다. 좋은 성능을 달성하기 위하여 상대적으로 용량이 작은 웃준위기억기를 요구한다면 그때 2준위기억기의 비트당 평균가격은 아래준위기억기의 가격과 근사하게 될것이다.

제 5 장. 외부기억기

◆◆◆

◆ 자기디스크는 여전히 외부기억기의 가장 중요한 구성부분이다. 넣었다꺼냈다할수 있는 디스크와 고정된 디스크(하드디스크)들은 개인용콤퓨터로부터 대형콤퓨터와 슈퍼콤퓨터에 이르기까지의 모든 체계에서 쓰인다.

◆ 더 큰 성능과 높은 리용성을 얻기 위하여 봉사기와 대형체계상에서 일반적으로 쓰이는 방식은 RAID(Redundant Array of Independent Disks)디스크기술이다. RAID 는 자료기억장치들을 병렬로 배렬하여 다중디스크들을 리용하기 위한 기술들의 집합이라고 할수 있다. 이것은 디스크고장을 보상하기 위하여 약간 여유 있게 설치한다.

◆ 빛기억기기술은 모든 형태의 콤퓨터체계에서 아주 중요하게 제기된다. 한편 CD-ROM 은 오래동안 널리 리용되여 왔고 오늘날에는 쓰기할수 있는 CD 와 자기빛기억기(MO)와 같은 현대기술들이 더욱더 중요한것으로 제기되고 있다.

◆◆◆

이 장에서는 외부장치와 체계들의 범위를 고찰한다. 제 1 절에서는 가장 중요한 장치인 자기디스크부터 고찰한다. 자기디스크는 사실상 모든 콤퓨터체계에서 외부기억기의 기본을 이룬다. 다음절은 RAID 라고 하는 체계계렬로 특별히 찾아 볼수 있는 보다 높은 성능을 가진 디스크배렬들의 리용을 고찰한다. 많은 콤퓨터체계에서 더욱더 중요한 구성부분은 외부빛기억기이며 이것은 제 3 절에서 보기로 한다. 마지막으로 자기테프에 대하여 서술하였다.

제1절. 자기디스크

디스크는 자성물질로 피복한 금속 또는 수지로 된 원판이다. 자료는 **자두**라고 하는 유도성선륜을 거쳐서 디스크에 기록할수도 있고 읽을수도 있다. 읽기나 쓰기조작할 때 자두는 멈춰 져 있으며 그때 원판은 그아래에서 회전한다.

쓰기는 선륜을 통해 흐르는 전류에 의하여 자기마당이 만들어 진다는 원리에 기초하고 있다. 임풀스는 자두에 전송되며 자기적패턴은 양전기와 음전기에 따라 서로 다른 패턴들로 되면서 아래면에 기록된다. 읽기는 선륜에 대해 상대적으로 움직이는 자기마당이 선륜에 전류를 발생시킨다는 원리에 기초하고 있다. 디스크면이 자두밑을 지날 때 자두에는 이미 디스크면에 기록되여 있는 자기적패턴과 같은 극성의 전류가 발생한다.

1. 자료구성과 형식

자두는 그아래에서 돌아 가는 원판의 부분에 대한 읽기나 쓰기능력을 가진 비교적 작은 장치이다. 이것은 **자리길**이라고 불리우는 고리들의 동심원묶음으로 원판우에 자료를 구성한다. 매 자리길은 자두와 같은 폭을 가진다. 이 폭은 일반적으로 500～2000 자리길/면이다.

그림 5-1 은 이 자료기록형식을 보여 준다. 린접자리길들은 **간격**에 의해 구분된다. 이것은 자두의 제작상 오유나 자기마당들의 호상간섭에 의한 장애를 막거나 될수록 작게 하자는데 있다. 전자기구를 간소화하기 위하여 일반적으로 매 자리길에 같은 수의 비트를 기억시킨다. 따라서 인치당 비트수로 지적되는 **밀도**는 제일 바깥자리길로부터 제일 안쪽자리길까지 이동하면서 증가한다.

자료는 **블로크**단위로 디스크에로 혹은 디스크로부터 전송된다. 일반적으로 블로크는 자리길의 용량보다 작다. 대체로 자료는 **분구**라고 하는 블로크크기로 기억된다(그림 5-1) . 일반적으로 자리길당 분구수는 10~100 개의 범위이며 고정 혹은 가변길이형식중의 하나로 되여 있다. 체계에서 특별히 지나친 밀도요구를 피하기 위하여 린접한 분구들은 분구들사이의 간격들로 구분된다.

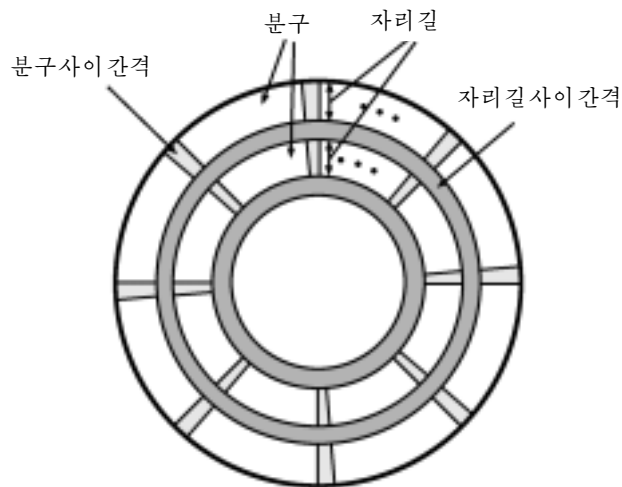



그림 5-1. 디스크자료배치도

일부 방법들은 자리길안의 분구위치들을 찾아 낼것을 요구한다. 명백한것은 자리길의 어떤 시작점이 있어야 하며 매 분구의 시작과 끝을 지적하는 방법이 있어야 한다는것이다. 이 요구들은 디스크에 기록된 조종자료에 의하여 실현된다. 따라서 디스크는 디스크구동에만 리용되는 몇개의 확장자료들로 형식화되며 사용자는 이 자료를 호출하지 못한다.

디스크형식화의 실례를 그림 5-2 에 주었다. 이 경우에 매개 자리길은 매개 분구가 600byte 로 된 30 개의 고정길이분구를 가진다. 매개 분구는

512byte의 자료와 디스크조종기에 필요한 조종정보를 가진다. ID마당은 개별적인 분구들을 배렬하기 위하여 리용되는 유일한 지적자 혹은 주소이다. 동기바이트는 마당의 시작을 지정하는 특수한 비트렬이다. 자리길번호는 면우의 자리길을 가리킨다. 디스크가 여러개의 면을 가지므로 자두번호는 자두를 지적한다. ID와 매개 자료마당들은 오유검사부호를 가진다.

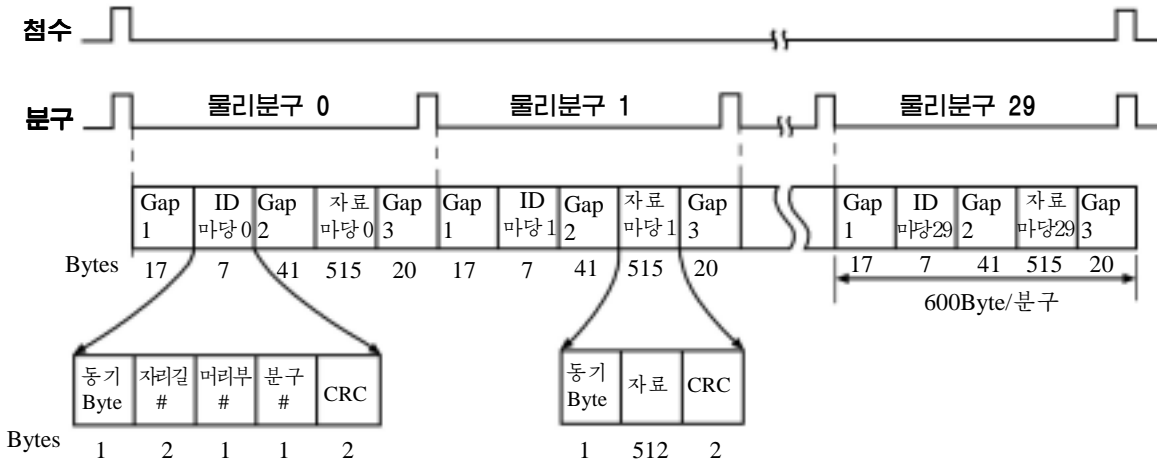



그림 5-2. 윈체스터디스크의 자리길형식(Seagate ST506)

2. 물리적특성

표 5-1에는 자기디스크의 여러가지 형태들을 구별하는 기본특성들을 주었다. 첫째로, 자두는 원판의 반경방향으로 고정되거나 이동할수 있어야 한다. **고정자두디스크**에서는 자리길마다에 읽기-쓰기자두가 하나씩 있다. 모든 자두는 자리길에 가로 놓인 든든한 지지대에 고정된다(그림 5-3 ㄱ). **이동자두디스크**에서는 읽기-쓰기자두 한개만 있다. 역시 자두는 지지팔에 설치된다(그림 5-3 ㄴ). 자두가 임의의 자리길우에 놓일수 있어야 하므로 지지팔은 이를 위하여 자리길의 가로방향으로 이동할수 있어야 한다.

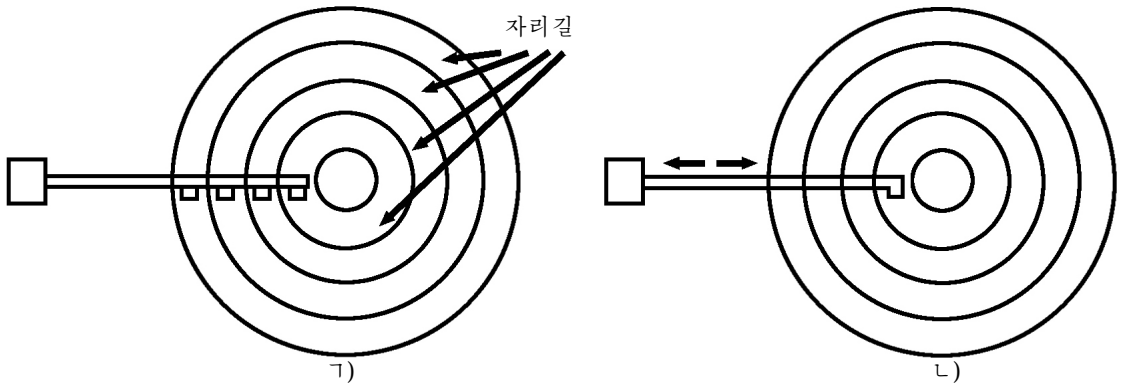



그림 5-3. 고정식과 이동식자두디스크
ㄱ-고정식자두, ㄴ-이동식자두

디스크자체는 팔과 디스크를 회전시키는 기구 그리고 자료를 입출력하는데 필요한 전자장치들로 구성된 디스크구동기에 설치된다. **하드디스크**는 디스크구동기에 영구적으로 설치된다. **넣었다꺼냈다할수 있는 디스크(플로피디스크)**는 구동기에서 꺼낼수 있으며 다른 디스크로 바꿀수 있다. 플로피디스크의 우점은 제한된 디스크체계들에서 많은 자료들을 리용할수 있다는것이다. 그러므로 이런 디스크들은 한 콤퓨터체계로부터 다른 콤퓨터체계에로 옮겨 갈수 있다.

표 5-1. 디스크체계의 물리적특성

자두이동	**원판수**
고정자두식(자리길 당 하나)	단일원판
이동자두식(면 당 하나)	다중원판
디스크휴대성	**자두동작**
비제거형디스크	접촉형(유연성자기디스크)
제거형디스크	고정간격
면수	공기부력식간격(윈체스터)
단일면식	
2 중면식	

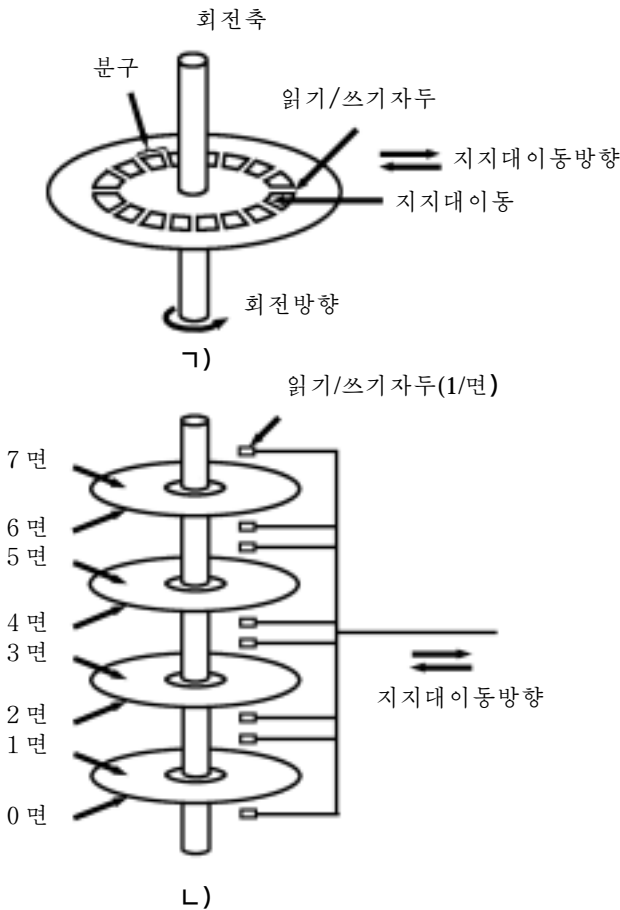



그림 5-4. 다중원판디스크

대부분의 디스크들에는 자화층이 **량면**원판의 두면에 입혀 져 있다. 일부 값이 눅은 디스크체계들에서는 **단면**디스크를 리용한다.

일부 디스크구동기들은 인치정도의 간격으로 수직으로 겹쌓인 **여러개의 원판**들을 조정한다(그림 5-4). 여기에는 여러개의 자두팔들이 설치된다. 원판들은 **디스크묶음**이라고 하는 장치에 설치된다.

마지막으로 자두기구는 세가지 형태로 디스크들을 분류한다. 전통적으로 읽기-쓰기자두는 원판우에 일정한 간격을 두고 위치한다. 또 다른 하나는 읽기/쓰기 를 조작하면서 매체에 물리적으로 접속되는 자두기구이다. 이 기구는 작고 유연하며 가장 값이 눅은 디스크형태인 **플로피디스크**에 리용된다.

디스크의 세번째 형식을 리해하기 위하여서는 자료밀도와 원판과 자두의 공간크기사이의 관계를 아는것이 필요하다. 자두가 정확히 쓰거나 읽기 위해서는 충분히 큰 전자기마당을 발생시키거나 수감할수 있어야 한다. 간격이 좁은 자두일수록 자기

기능을 수행하기 위하여 원판면에 더 가까이 접근해야 한다. 더 좁은 자두일수록 자리길을 더 좁게 하며 따라서 바라는 자료밀도는 더 커진다. 그러나 자두가 원판에 접근할수록 불순물이나 설치의 불합리성으로 인한 오유발생률이 커진다. 기술이 발전함에 따라 윈체스터디스크가 개발되였다. 윈체스터자두는 거의 먼지가 없는 밀봉된 구동기묶음들에 리용된다. 이것들은 전통적인 고정디스크자두보다 디스크의 면에 더 가까이 접근하도록 설계되였다. 따라서 자료밀도가 아주 크다. 자두는 디스크가 움직이지 않을 때 실제로 원판면우에 가볍게 놓여 있는 기체력학식금속판이다. 회전하는 디스크에 의하여 일어 나는 공기의 부력은 원판면우에서 금속판을 띄우는데 충분하다. 이와 같은 비접촉체계는 전통적인 하드디스크자두보다 원판면에 자두를 더 가까이 접근시킬수 있다.[1]

3. 디스크성능지표

디스크 I/O 조작의 실제적인 세부들은 콤퓨터체계 즉 조작체계, I/O 통로의 성질, 디스크조종장치에 의존한다. 디스크 I/O 전송의 일반적인 시간선도를 그림 5-5에 보여 주었다.

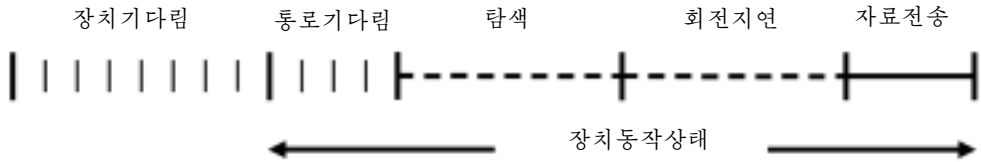



그림 5-5. 디스크 I/O 전송의 시간선도

디스크구동기가 동작하고 있을 때 디스크는 정상적인 속도로 회전한다. 읽기나 쓰기를 하기 위하여 자두는 요구하는 자리길과 그 자리길의 요구하는 분구에 설치되여야 한다. 자리길선택은 이동식자두체계인 경우에 자두를 이동시키고 고정자두체계인 경우에 해당 자두를 전기적으로 선택하는 동작을 진행한다. 자리길에 자두를 설치하는데 걸리는 시간을 **탐색시간**이라고 한다. 어떤 경우에는 일단 자리길이 선택되면 디스크조종기는 해당한 분구가 자두와 일치하는 선상에서 되돌아 올 때까지 기다린다. 분구의 시작부분이 자두에 도착할 때까지의 시간을 **회전지연**이라고 한다. **탐색시간**과 회전지연의 합이 **호출시간**이다. 이 시간이 바로 읽기나 쓰기를 위한 위치를 차지할 때까지 걸리는 시간이다. 일단 자두가 위치를 차지하면 그다음 읽기나 쓰기조작은 분구가 자두밑을 통과함으로써 수행된다. 이것이 조작의 자료전송부분이다.

일반적으로 호출시간과 전송시간외에 디스크 I/O조작과 관련된 여러가지 대기지연이 있다. 처리공정이 I/O 요구를 발생할 때 우선 이 요구를 리용하기 위한 장치는 대기렬에서 기다려야 한다. 그다음 장치는 처리공정에 할당된다. 만일 장치가 단일 I/O 통로나 I/O 통로들의 묶음과 다른 디스크구동기들을 공유한다면 그때 리용할수 있는 통로에 대하여

---

[1] 력사적으로 흥미 있는 Winchester 라는 말은 이 디스크가 개발되기전에 3340 디스크형에서 코드의 이름으로 IBM 이 리용하였다. 3340 은 묶음단에 밀페식자두를 가진 이동식디스크묶음이였다. 이 용어는 현재 기체력학식자두설계로 만든 임의의 밀봉된 디스크구동기에 적용되고 있다. Winchester 디스크는 개인용콤퓨터와 워크스테이션에 설치되며 거기서는 **하드디스크**라고 한다.

추가적인 기다림이 생길수 있다. 이 점에서 디스크호출을 시작하기 위하여 탐색을 진행한다.

일부 대형콤퓨터체계에서 순환위치수값(RPS)이라고 하는 기술을 리용한다. 이것은 다음과 같이 동작한다. 탐색명령이 통과되였을 때 통로는 다른 I/O 조작을 하기 위하여 해방된다. 탐색이 완성되였을 때 장치는 언제 자료가 자두밑에 돌아 오는가를 확정한다.

해당 분구가 자두에 접근하면 장치는 주콤퓨터에로 통신로를 회복하도록 한다. 만일 조종장치나 I/O 통로가 다른 I/O 조작상태에 있으면 그때 결합시도는 실패로 되며 장치는 RPS 실패라고 하는 재결합을 시도하기전에 전체순환을 바꾸어야 한다. 이것은 그림 5-5에서 시간축에 더 추가되는 확장된 지연요소이다.

탐색시간

탐색시간은 요구하는 자리길에로 디스크지지대를 이동시키는데 요구되는 시간이다. 이 시간은 고정시키기 어려운 량이다. 탐색시간은 여러가지 기본구성요소들로 이루어 진다. 즉 초기시동시간, 일단 호출지지대가 속도를 높이면 교차해야 할 자리길들을 가로지르는데 걸리는 시간이다. 그러나 가로 지르는데 걸리는 시간은 자리길수와 선형관계를 가지지 않는다. 다음의 1 차식으로 근사화할수 있다.

$$T_s = m \times n + s$$

여기서 T_s – 평가하려는 탐색시간,
n – 가로 지른 자리길수,
m – 디스크구동기에 의존하는 상수,
s – 시동시간

실례로 개인용콤퓨터에서 값이 눅은 하드디스크는 m=0.3ms, *s*=20ns 정도이다. 한편 더 크고 비싼 디스크구동기는 m=0.1ms, s=3ms 이다.

순환지연

유연성자기디스크와 다른 디스크들인 경우 일반적으로 분당 회전수는 3000r/ min 이며 따라서 1 회의 회전시간은 16.7ms 이다. 그러므로 평균지연은 100~200ms 사이에 있다.

전송시간

디스크에 대한 전송시간은 다음식에서와 같이 디스크의 회전속도에 의존한다.

$$T = \frac{b}{rN}$$

여기서 T – 전송시간, b – 전송되는 바이트수,
N – 자리길당 바이트수, r – 회전속도, r /s

따라서 총적인 평균호출시간은 다음과 같이 결정할수 있다.

$$T_a = T_s + \frac{1}{2r} + \frac{b}{rN}$$

여기서 T_s –평균탐색시간

시간비교

앞에서 정의한 파라메터들에 대하여 평균값들에 대한 의존위험성을 보여 주는 서로 다른 두가지 I/O 조작들을 고찰해 보자. 평균탐색시간이 20ms, 전송속도가 1Mbyte/s, 한개 자리길이 32 개 분구이고 매 분구가 512byte 인 일반디스크를 고찰하자. 256 개의 분구로 구성된 총 128KB 의 파일을 읽으려고 한다고 가정하자. 이때 전송을 위한 총 시간을 계산해 보자.

우선 파일이 디스크에 조밀하게 기억되여 있다고 가정한다. 즉 파일은 8 개의 린접 자리길(8 자리길 × 32 분구/자리길=256 분구)의 모든 분구를 차지한다. 이것을 **순차적구성**이라고 한다. 이때 첫번째 자리길을 읽는 시간은 다음과 같다.

평균탐색	20.0	ms
회전지연	8.3	ms
32 분구읽기	16.7	ms
	45	ms

나머지자리길들은 실제적으로 탐색시간없이 읽기할수 있다. 즉 I/O 조작은 디스크의 이동(회전)으로 유지할수 있다. 이때 대부분의 련속적으로 뒤따르는 매개 자리길에 대하여 회전지연만이 필요하다. 따라서 매개 련속적인 자리길은 8.3 × 16.7=25ms 로 읽기된다. 전체 파일을 읽는데는 다음과 같은 시간이 걸린다.

총 시간 = 45 + 7 × 25 = 220ms = 0.22s

이제 순차적호출보다 자유호출을 리용하는 경우 같은 자료를 읽는데 요구되는 시간을 계산하자. 즉 디스크우에 임의로 분산된 분구들을 호출한다. 매 분구에 대하여

평균탐색	20.0	ms
회전지연	8.3	ms
32 분구읽기	0.5	ms
	28.8	ms

총 시간= 256 × 28.8= 7373ns= 7.37s

이다.

총 분구들이 디스크로부터 읽어 지는 순서가 I/O 성능에 큰 영향을 준다는것은 명백하다. 여러 분구들이 읽거나 써지는 파일을 호출하는 경우에 자료의 분구들을 전개하는 방법에 대한 몇가지 조종이 있으며 다음장에서 이 문제를 설명하도록 한다. 그러나 파일호출의 경우에도 다중프로그람환경에서 같은 디스크에 대한 경쟁을 요구하는 I/O 들이 있을수 있다. 따라서 디스크 I/O 의 성능이 디스크에 대한 순전히 우연적인 호출로 이루어 지는 경우 개선할수 있는 방법을 조사하여야 한다. 이것은 이 책에서 고찰하는 범위가 아니며 조작체계의 범위인 디스크순서짜기알고리듬의 고찰에 귀착된다([STAL98]을 참고).

제 2 절. RAID

이미 론의된바와 같이 2 차기억기성능의 발전속도는 처리장치와 주기억기의 속도보다 현저하게 뜨다. 이 불균형으로부터 전반적인 콤퓨터체계성능을 개선하는데서 주되는 초점으로 되는것은 대체로 디스크기억장치체계이다.

콤퓨터성능의 다른 범위에서처럼 디스크기억장치설계자는 지금까지 한개 디스크구성에만 힘을 넣었지만 더 높은 성능을 얻자면 다중병렬구성을 리용해야 한다는것을 인식하였다. 디스크기억장치의 경우에 이것은 독립적으로 그리고 병렬로 조작하는 디스크배렬구성을 개발하게 한다. 다중디스크에서 갈라 진 디스크들에 요구하는 자료들이 오래동안 상주하므로 갈라 진 I/O 요구들은 병렬로 조종될수 있다.

다중디스크들의 리용과 함께 자료를 구성하는 여러가지 방법들이 있으며 이 방법들에서는 믿음성을 개선하기 위하여 여유를 추가할수 있다. 이것은 여러개의 가동환경과 조작체계들에서 리용할수 있는 자료기지를 개발하는것을 저애한다. 다행스럽게도 산업분야에서는 RAID 라고 하는 다중디스크자료기지의 규격화된 설계방식에 대한 의견일치를 보았다. RAID 방식은 0~6 까지 7 개 준위로 이루어 졌다.[2] 이 준위들은 계층적관계를 의미하는것이 아니라 3 개의 공통적인 특성들을 공유하는 여러가지 설계방식들을 지적한것이다. 이 3 가지 공통적인 특성들은 다음과 같다.

- RAID 는 조작체계가 단일국부구동기로서 리해하는 물리적디스크구동기들의 묶음이다.
- 자료는 물리적구동기배렬전체에 분포된다.
- 여유디스크용량은 기우성정보를 기억하는데 리용되며 이것은 디스크에 오유가 생긴 경우에 자료의 회복을 담보한다.

두번째와 세번째 특성들의 세부내용은 서로 다른 RAID 준위들에서 차이난다. RAID0 은 세번째 특성을 지원하지 못한다.

RAID 라는 말은 처음에 캘리포니아 버클리종합대학의 한 연구집단에 의하여 발표되였다[PATT88].[3] 론문에는 여러가지 RAID 구성과 응용을 소개하였으며 지금까지 리용되고 있는 RAID 준위들을 정의하였다. RAID 전략은 큰 용량의 디스크구동기를 다중화된 보다 작은 용량구동기들로 바꾸고 다중구동기로부터 자료를 동시에 호출할수 있는 방법으로서 자료를 분리하였다. 그리하여 성능이 개선되고 용량효률을 높이였다.

RAID 제안의 유일한 공헌은 파잉에 대한 요구를 효과적으로 론의한것이다. 비록 동시에 동작하는 여러개의 자두와 수행기구들을 허락하는것이 더 높은 성능과 전송속도를 달성하지만 여러개 장치들을 리용하면 일반적으로 오유가 커진다. 믿음성이 작아 지는것을 보상하기 위하여 RAID 는 디스크오유로 잃어 버리는 자료를 복귀하도록 하는 기억된 기우성정보를 리용한다.

이제 RAID 의 매개 준위를 고찰하자. 표 5-2 에 7 개 준위들을 개괄하였다. 물론 2준위와 4준위는 상품화되지 못했으며 공업적으로 실현하는데 적합지 않다. 그럼에도 불구하고 이 준위들에 대한 해설은 일부 다른 준위들에서의 설계선택을 도와 준다.

그림 5-6 은 여유용량이 없는 4 개 디스크를 요구하는 자료용량을 지원하기 위하여 7 개의 RAID 를 리용한 실례이다. 그림은 사용자자료와 여유자료를 구분하여 표시하였으며 관계되는 여러 준위들에서의 기억기요구를 지적하였다. 다음론의를 통하여 이 그림을 따져 보자.

---

[2] 추가적인 준위들은 일부 연구사들과 일부 회사들에 의하여 정의되였으나 이 절에서 서술한 7 개 준위들은 일반적으로 합의한것이다.

[3] 이 론문에서 략자 RAID 는 Redundant Array of Inexpensive Disks (비용이 적은 디스크들의 여유배렬)로 표기된것이다. **비용이 적게 든다**는 말은 하나의 큰 비용이 드는 디스크(SLED)대신에 RAID 에 상대적으로 비용이 적게 드는 작은 디스크들을 대비적으로 리용하였다. SLED 는 물질적으로 RAID 와 비RAID 구성들에서 리용되고 있는 류사한 디스크기술로 된 이전의 기억기들이다. 그러므로 산업에서는 RAID 배렬이 충분한 성능과 믿음성의 증대를 강조한 **독립성**이라는 말을 써왔다.

표 5-2. RAID 준위

종류	준위	설명	I/O 요구속도 (읽기/쓰기)	자료전송속도 (읽기/쓰기)	대표적인 응용
렬화	0	여유디스크없음	큰 렬: 가장 빠름	작은 렬: 가장 빠름	부차적인 자료에 대한 고성능을 요구하는 응용프로그람
반사성	1	여유디스크에 자료반사보관	빠름/중간	느림/느림	체계구동장치:중요자료
병렬호출	2	여유디스크에 하밍부호보관	느림	가장 빠름	
	3	여유디스크에 비트간 기우성비트보관	느림	가장 빠름	화상처리, CAD 와 같이 방대한 I/O 처리를 요구하는 응용프로그람
독립호출	4	여유디스크에 블로크준위기우성블로크보관	가장 빠름/중간	중간/느림	
	5	여유디스크에 블로크준위분산형기우성블로크보관	가장 빠름/중간	중간/느림	높은 요구속도, 읽기강조, 자료의 탐색
	6	여유디스크는 블로크준위분산형 2 중기우성블로크보관	가장 빠름/느림	중간/느림	대단히 높은 가동능률을 요구하는 응용프로그람

1. RAID 준위 0

RAID 준위 0 은 성능개선을 위한 여유용량을 포함하지 않으므로 RAID 계렬의 진짜 성원이 아니다. 그러나 성능과 용량이 1 차적관심사로 되며 가격을 눅게 하는것이 믿음성을 높이는것보다 더 중요하게 제기되는 일부 슈퍼콤퓨터들에 약간 적용되고 있다.

RAID 0 에 있어서 사용자와 체계의 자료들은 배렬형식으로 모든 디스크들에 분포된다. 이것은 하나의 큰 디스크를 리용하는 경우에 비하여 많은 우점을 가진다. 즉 두개의 서로 다른 I/O 요구가 서로 다른 두개의 자료블로크들을 기다리고 있다면 그때에 요구되는 블로크들이 서로 다른 디스크에 있게 되는 좋은 기회가 생긴다. 따라서 두 요구는 I/O 기다림시간을 줄이면서 병렬로 처리될수 있다.

그러나 모든 RAID 준위들에서와 마찬가지로 RAID 0 은 디스크배렬형식으로 자료를 간단히 분포하는것보다 더 진행된다. 즉 자료는 리용할수 있는 디스크들에서 줄을 짓는다. 이것은 그림 5-7 을 고찰하면 아주 쉽게 리해된다. 모든 사용자와 체계의 자료들은 론리적디스크에 기억된것으로 볼수 있다. 디스크는 줄별로 분할된다. 즉 이 줄들은 물리적인 블로크, 분구, 기타 단위들일수 있다. 줄들은 련속적인 배렬성원들에 순환고리를 그려 준다. 매개 배렬성원들에게 한개의 줄을 정확히 그려 주는 론리적인 련속줄묶음을 **줄무늬**라고 한다. n 개의 디스크배렬에서 첫 n 개의 론리적줄들은 물리적으로 매 n 개 디스크상에 첫 줄들로 기억되여 첫 줄무늬를 형성한다. 즉 두번째 n 개의 줄들은 매개 디스크에 두번째 줄들로서 기억된다. 이 배렬의 우점은 가령 한개의 I/O 요구가 론리적으로 린접한 다중줄들로 구성된것보다 더 크다면 요구된 n 개이상의 줄들을 병렬로 조종하여 I/O 전송시간을 훨씬 단축할수 있는것이다.

쪼각 0
쪼각 1
쪼각 2
쪼각 3
쪼각 4
쪼각 5
쪼각 6
쪼각 7
쪼각 8
쪼각 9
쪼각 10
쪼각 11
쪼각 12
쪼각 13
쪼각 14
쪼각 15
ㄱ)
쪼각 0
쪼각 1
쪼각 2
쪼각 3
쪼각 0
쪼각 1
쪼각 2
쪼각 3
쪼각 4
쪼각 5
쪼각 6
쪼각 7
쪼각 4
쪼각 5
쪼각 6
쪼각 7
쪼각 8
쪼각 9
쪼각 10
쪼각 11
쪼각 8
쪼각 9
쪼각 10
쪼각 11
쪼각 12
쪼각 13
쪼각 14
쪼각 15
쪼각 12
쪼각 13
쪼각 14
쪼각 15
ㄴ)
비트 0
비트 1
비트 2
비트 3
검사비트0
검사비트1
검사비트2
ㄷ)
비트 0
비트 1
비트 2
비트 3
기우성비트
ㄹ)
블로크 0
블로크 1
블로크 2
블로크 3
기우성블로크 (0-3)
블로크 4
블로크 5
블로크 6
블로크 7
기우성블로크 (4-7)
블로크 8
블로크 9
블로크 10
블로크 11
기우성블로크 (8-11)
블로크 12
블로크 13
블로크 14
블로크 15
기우성블로크 (11-15)
ㅁ)

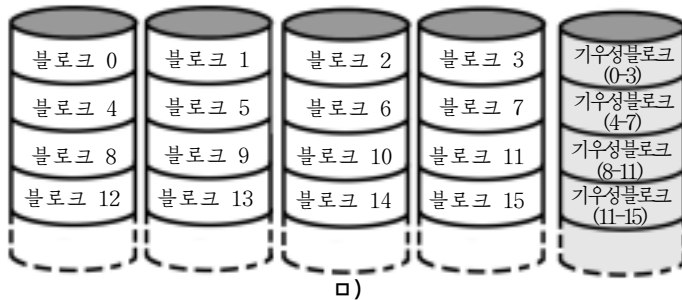
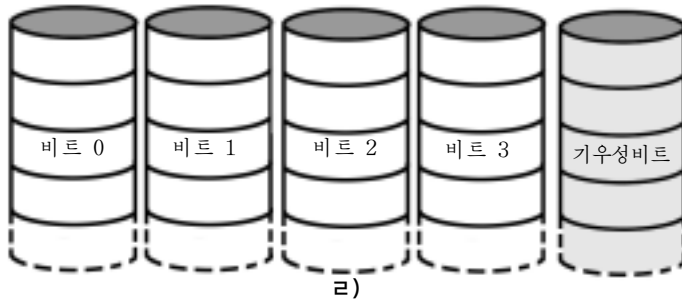
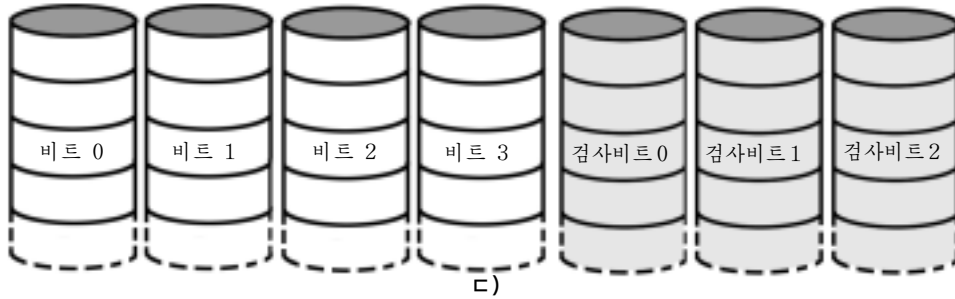
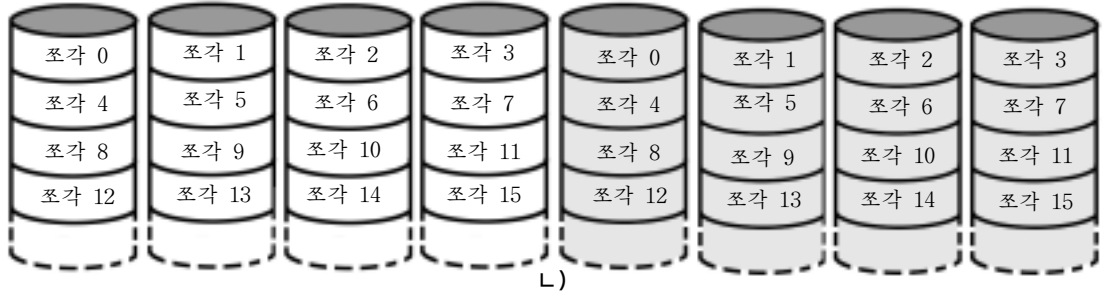
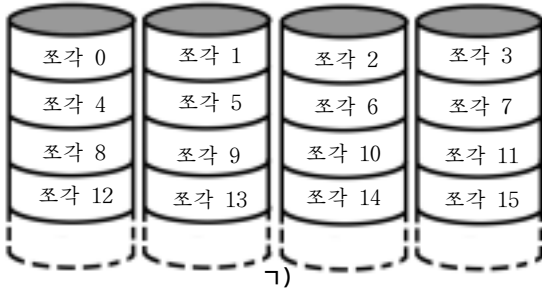
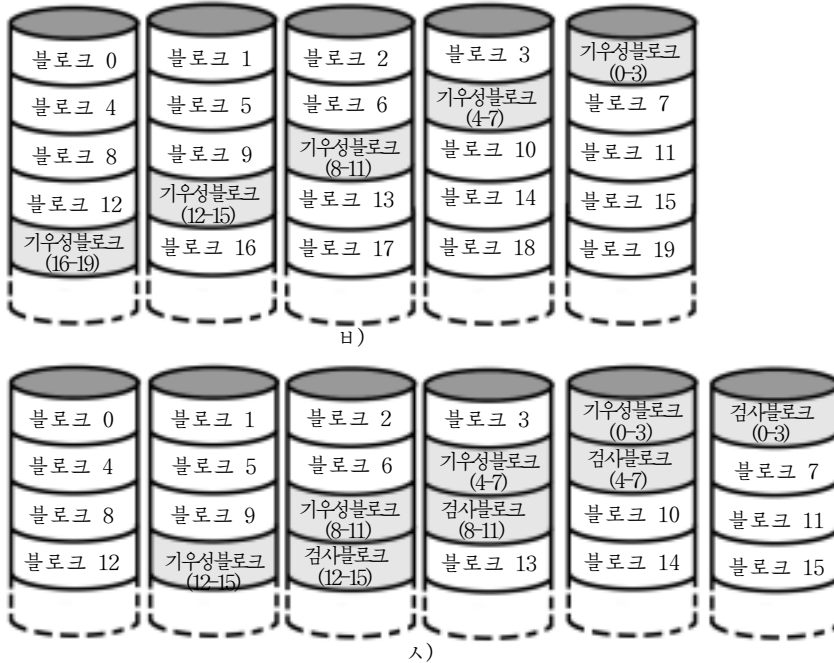



그림 5-6. RAID 준위
ㄱ-RAID 0(초파디스크 없음), ㄴ-RAID 1(초파디스크는 자료반사보관), ㄷ-RAID 2(초파디스크는 하밍부호보관), ㄹ-RAID 3(초파디스크는 비트간 기우성비트보관), ㅁ-RAID 4(초파디스크는 블로크준위기우성블로크보관), ㅂ-RAID 5(초파디스크는 블로크준위 분산형기우성블로크보관), ㅅ-RAID 6(2중초파디스크리용)

그림 5-7에는 론리적디스크공간과 물리적디스크공간사이에 배치하기 위한 배렬관리프로그람의 리용을 보여 주었다. 이 프로그람은 디스크보조체계 혹은 주콤퓨터들에서 집행할수 있다.

높은 자료전송능력을 위한 RAID 0

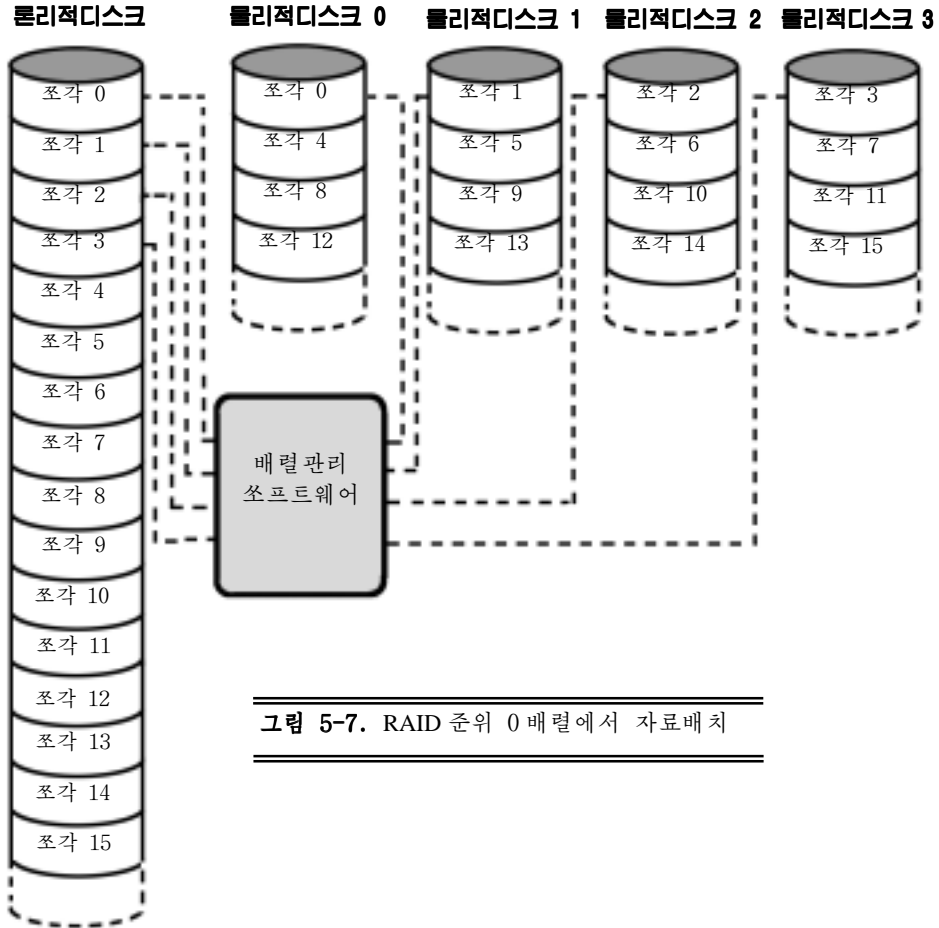
모든 RAID 준위들의 성능은 주체계의 요구페턴들과 그리고 자료의 편성에 결정적으로 의존한다. 이 지표들은 여유량의 영향이 분석과 서로 간섭하지 않는 RAID 0에서 가장 명백하게 찾아 볼수 있다. 첫째로, 높은 자료전송속도를 얻기 위한 RAID 0의 리용을 고찰한다. 높은 전송속도를 보기 위한 응용인 경우 두가지 요구가 제기된다. 첫째로, 높은 전송능력은 주기억기와 개별적인 디스크구동기들사이의 전반적인 경로를 통하여 실현되여야 한다. 이것은 내부조종기모선, 주체계 I/O 모선, I/O 접속기, 주기억기모선을 포함한다.

둘째로, 응용이 디스크배렬을 효과적으로 구동하는 I/O 요구들을 만들어 주어야 한다. 이 요구는 일반적요구가 렬의 크기에 비하여 론리적으로 련속적인 자료들의 량이 큰 경우에만 제기된다. 이 경우에 단일 I/O명령요구는 여러개의 디스크들로부터 병렬자료전송에 의해 초래되며 단일디스크전송에 비하여 전송속도와 효률이 높다.

높은 I/O 요구속도를 위한 RAID 0

동작적응환경에서 사용자는 일반적으로 전송속도보다도 응답시간에 더 관심을 가진

다. 작은 자료량에 대한 개별적 I/O 요구인 경우 I/O 시간은 디스크자두의 운동(탐색시간)과 디스크의 이동(회전지연)에 의하여 결정된다.

그림 5-7. RAID 준위 0 배렬에서 자료배치

동작환경에서 초당 수백개의 I/O 요구들이 제기될수 있다. 디스크배렬은 여러개의 디스크들에 I/O 부하를 평균화하여 분포함으로써 높은 I/O 집행속도를 얻을수 있다. 효률적인 부하평균화는 일반적으로 다중 I/O 의 독립적인 응용 혹은 여러개의 비동기적인 I/O 요구를 가질수 있는 단일동작적응환경이라는것을 암시한다. 성능은 또한 줄의 크기의 영향도 받는다. 줄의 크기가 비교적 크므로 한개의 I/O 요구는 오직 하나의 디스크호출만 가지며 다른 한편 여러개의 I/O 요구기다림은 병렬로 조종되여 매 요구에 대한 기다림시간이 줄어 들게 된다.

2. RAID 준위 1

RAID 1 은 여유용량을 가지고 있는 방법인 2~6 까지의 RAID 준위와는 차이난다. RAID 1 에서 여유용량이 전체 자료를 2중화하는 간단한 방법으로 리용되는것과 마찬가지로 다른 RAID 방법들에서는 여유를 도입하기 위하여 일부 기우성계산의 형식들이 리용된다면 RAID 1 에서는 여유가 모든 자료들을 복재하는 간단한 방책에 의하여 이루어 진

다. 그림 5-6 ㄴ에서 보여 주는바와 같이 RAID 0 에서처럼 자료를 줄로 배치하는 방법이 리용되였다. 그러나 이 경우에 모든 론리적줄들은 두개의 물리적디스크들에 배치되므로 배렬된 매개 디스크들은 같은 자료를 포함하는 복사(반사)디스크를 가진다.

RAID 1 구성에는 다음과 같은 여러가지 긍정할만한 측면들이 있다.

- 읽기요구는 요구하는 자료를 가지고 있는 두개의 디스크들중 어느 하나에 의하여 봉사될수 있으며 어느것이나 최소탐색시간과 최소회전지연을 가진다.
- 쓰기요구는 두개의 대응하는 줄들을 갱신할것을 요구하며 이것은 병렬로 진행될수 있다. 따라서 쓰기성능은 느린 두개의 쓰기동작 (실례로 긴 탐색시간과 지연시간을 가지는) 으로 주어 진다. 그러나 RAID 1 에는 《쓰기반칙》이 없다. RAID 준위 2~6 은 기우성비트를 리용한다. 그러므로 한개의 줄이 갱신될 때 배렬관리프로그람은 제기된 실제줄을 갱신할 때에 먼저 기우성비트들을 계산하고 갱신한다.
- 오유에 대한 보정은 간단하다. 구동기에서 오유가 발생되였을 때 자료는 두번째 구동기로부터 호출될수 있다.

RAID 1 의 기본결함은 값이 비싼것이다. 즉 RAID 1 은 자기를 지원하는 론리적디스크의 2 배에 달하는 디스크공간을 요구한다. 왜냐하면 RAID 1 구성은 체계프로그람과 자료 그리고 다른 아주 중요한 파일들을 보관하는 구동기들에서 제한을 받게 되기때문이다. 이 경우에 RAID 1 은 모든 자료들에 대한 실시간여벌복사를 보장함으로써 디스크오유가 있는 경우에 모든 중요한 자료들을 여전히 즉시에 리용할수 있게 한다.

동작지향환경에서 RAID 1 이 많은 I/O 요구들을 읽는다면 높은 I/O 요구속도를 달성할수 있다. 이 경우에 RAID 1 의 성능은 RAID 0 의 성능의 거의 2 배에 도달할수 있다. 그러나 만일 I/O 요구들의 중요한 부분이 쓰기요구를 가진다면 이때 RAID 0 에서는 충분한 성능을 얻지 못할수 있다. RAID 1 은 또한 높은 읽기의 비률을 가진 자료전송이 많은 응용프로그람에 있어서 RAID 0 보다 개선된 성능을 보장할수 있다. 만일 응용프로그람이 매개 읽기요구를 분할하여 두개의 디스크성원들이 참가하도록 한다면 성능이 개선된다.

3. RAID 준위 2

준위 2 와 3 의 RAID 는 병렬호출기능을 리용한다. 병렬호출배렬에서 모든 디스크들은 다 I/O 요구들의 집행에 참가한다. 일반적으로 개별적인 구동축들이 동기화되므로 모든 디스크자두들은 임의의 주어 진 순간에 매 디스크들의 같은 위치에 있다.

다른 RAID 방법들에서처럼 자료를 줄로 배렬하는 방법을 리용한다. 준위 2 와 3 의 RAID 인 경우에 줄들은 매우 작으며 흔히 1byte 혹은 한단어로 주어 진다. RAID 2 에서 오유수정부호는 매 자료디스크에서 해당하는 비트전체에 대하여 계산되며 부호의 비트들은 다중기우성디스크상의 해당하는 비트위치들에 기억된다. 일반적으로 단일비트오유들을 수정하고 2 중비트오유를 검사하는데 하밍부호가 리용된다.

비록 RAID 2 가 RAID 1 보다 디스크수를 적게 요구하지만 여전히 비용이 많이 든다. 여유디스크수는 자료디스크수의 로그에 비례한다. 한번의 읽기에서 모든 디스크들은 동시에 호출된다. 요구된 자료와 관련되는 오유수정부호는 배렬조종기에 전송된다. 만일 단일비트오유가 있다면 조종기는 즉시에 응답하고 오유를 수정할수 있으며 따라서 읽기호출시간은 떠지지 않는다. 한번의 쓰기에서 모든 자료디스크들과 기우성디스크들이 쓰기조작에 호출되여야 한다.

RAID 2 는 디스크오유가 많이 발생하는 환경에서만 효과적으로 리용될수 있다.

개별적인 디스크들과 디스크구동기들의 믿음성이 높으면 RAID 2 는 필요 없으며 실현하지 않는다.

4. RAID 준위 3

RAID 3 은 RAID 2 와 비슷한 형식으로 구성된다. 차이점은 RAID 3 이 하나의 여유디스크만을 요구한다는것이며 얼마나 큰 디스크배렬이 있어야 하는가 하는 문제는 제기되지 않는다는것이다. RAID 3 은 작은 줄들로 분할된 자료에 대하여 병렬호출을 채용한다. 오유수정부호대신에 간단한 기우성비트가 모든 자료디스크들의 같은 위치에서 개별적인 비트들의 묶음을 계산한다.

여유

장치오유인 경우에 기우성구동기가 호출되며 자료는 나머지장치들로부터 재구성된다. 일단 오유가 발생한 구동기가 교체되면 잃어 버린 자료는 새 구동기에 다시 기억시킬수 있으며 조작은 다시 진행된다.

자료의 재구성은 아주 간단하다. X0~X3 이 자료를 포함하고 X4 가 기우성디스크인 5 개 구동기의 배렬을 고찰하자. i 번째 비트에 대한 기우성은 다음과 같이 계산된다.

$$\mathrm{X4}(i) = \mathrm{X3}(i) \oplus \mathrm{X2}(i) \oplus \mathrm{X1}(i) \oplus \mathrm{X0}(i)$$

X1 구동기가 오유를 나타냈다고 가정하자. 만일 웃식의 량변에 $\mathrm{X4(i)} \oplus \mathrm{X1(i)}$를 더하면 다음과 같이 된다.

$$\mathrm{X1}(i) = \mathrm{X4}(i) \oplus \mathrm{X3}(i) \oplus \mathrm{X2}(i) \oplus \mathrm{X0}(i)$$

따라서 X1 에서 자료의 매개 줄들의 내용은 배렬에 있는 나머지디스크들에 있는 대응하는 줄들의 내용으로부터 재발생할수 있다. 이 원리는 RAID 준위 3~6 에서도 모두 같다.

디스크오유인 경우 모든 자료는 축소방식을 리용한다. 이 방식에서 읽기인 경우 잃어 버린 자료는 안맞음론리합계산에 의하여 재생된다. 자료가 축소된 RAID 3 배렬에 쓰기될 때 기우성생성은 후에 재생할수 있어야 한다. 충분한 조작이 되돌려 지자면 오유디스크가 재배치되고 오유디스크의 전체 내용이 새로운 디스크에 재생되여야 한다.

성능

자료가 대단히 작은 줄들로 분할되므로 RAID 3 은 대단히 높은 자료전송속도를 얻을수 있다. 임의의 I/O 요구는 모든 자료디스크로부터 병렬자료전송을 가질수 있다. 큰 전송들에서 성능개선은 특별히 뚜렷하다. 다른 한편 하나의 I/O 요구만은 단번에 집행할수 있다. 따라서 동작지향환경에서 성능은 무시될수 있다.

5. RAID 준위 4

RAID 준위 4~6 은 독립적인 호출기능을 리용하였다. 독립적인 호출배렬에서 매개 디스크부분들은 독립적으로 동작하므로 분할된 I/O 요구는 병렬조작을 만족할것이다. 이것으로 인하여 독립적인 호출배렬은 높은 I/O 속도를 요구하는 응용들에 더 적합할수 있으며 높은 자료전송속도를 요구하는 응용들에 상대적으로 적합하지 못하다.

다른 RAID 방법들에서와 마찬가지로 자료분할방법을 리용한다. RAID 준위 4~6 의

경우에 줄들은 상대적으로 크다. RAID4 에서 기우성비트배렬(bit-by-bit parity)렬은 매개 자료디스크에서 대응하는 줄들에 대하여 계산하며 기우성비트들은 기우성디스크상에 대응하는 줄로 기억된다.

RAID 4 는 작은 용량의 I/O 쓰기요구가 수행될 때 반칙을 가진다. 쓰기가 발생할 때마다 배렬관리프로그람은 사용자자료뿐아니라 대응하는 기우성비트들로 갱신되여야 한다. X0~X3 이 자료를 포함하고 X4 가 기우성디스크인 5 개 구동기의 배렬을 고찰하자. 쓰기가 디스크 X1 의 줄에 대해서만 수행된다고 가정한다. 처음에 매 비트에 대하여 다음과 같은 관계가 성립한다.

$$X4(i) = X3(i) \oplus X2(i) \oplus X1(i) \oplus X0(i)$$

갱신후에 될수록 변화된 비트들은 처음의 부호에 의하여 지적된다.

$$\begin{aligned} X4'(i) &= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \\ &= X3(i) \oplus X2(i) \oplus X1'(i) \oplus X0(i) \oplus X1(i) \oplus X1(i) \\ &= X4(i) \oplus X1(i) \oplus X1'(i) \end{aligned}$$

새로운 기우성을 계산하기 위하여 배렬관리프로그람은 낡은 줄들과 낡은 기우성줄들을 읽어야 한다. 그다음 새로운 자료와 새롭게 개선된 기우성을 가진 이 두개의 줄들을 갱신한다. 따라서 매개 줄쓰기는 두번의 읽기와 두번의 쓰기를 가진다.

모든 디스크들이 줄들을 모두 포함하는 더 큰 용량의 I/O 쓰기인 경우에 기우성은 새 자료비트들만을 리용하여 계산함으로써 쉽게 계산된다. 따라서 기우성구동기는 자료구동기들에서 병렬로 갱신할수 있으며 확장된 읽기나 쓰기는 없다.

임의의 경우에 모든 쓰기조작은 기우성디스크를 호출하며 따라서 성능제한현상이 나타날수 있다.

6. RAID 준위 5

RAID 5 는 RAID 4 와 류사한 형태로 구성된다. 차이는 RAID5 는 모든 디스크들에 기우성줄들을 분할한것이다. 표준적배치는 그림 5-6 ㅂ에서 지적된 순환고리방법이다. N 개의 디스크배렬에 대하여 기우성줄은 첫 n 개의 줄들에 대하여서도 다른 디스크들에 존재하며 그다음 패턴들은 반복된다.

모든 구동기에 대한 기우성분할은 RAID4 에서 나타나는 I/O 성능제한현상을 피하게 한다.

7. RAID 준위 6

RAID 6 은 머클리연구소의 소론문에 소개되였다[KAT89]. RAID 6 에서 두개의 서로 다른 기우성계산들은 서로 다른 디스크들에 분리된 블로크들을 전송하고 기억한다. 따라서 사용자자료가 N 개의 디스크들을 요구하는 RAID 6 배렬은 N+2 개의 디스크들로 구성된다.

그림 5-6 ㅅ에서 이 방법을 설명해 보자. P 와 Q 는 두개의 서로 다른 자료검사알고리듬들이다. 도중의 하나는 RAID 준의 4 와 5 에서 리용된 알맞음론리합계산이다. 그러나 다른 하나는 독자적인 자료검사알고리듬이다. 이것은 두개의 디스크가 사용자자료오유를 가질 때 자료를 재생할수 있게 한다.

RAID 6 의 우점은 아주 높은 자료리용률을 준다는것이다. 3 개의 디스크들은 자료가 리용될수 없게 하는 MTTR 시간안에 오유를 가질수 있다. 다른 한편 RAID 6 은 중요한 쓰기반칙을 나타낼수 있는데 그것은 매개 쓰기가 두개의 기우성블로크에 영향을 미치기 때문이다.

제 3 절. 빛기억기

1983 년에 일찌기 없었던 가장 성공적인 소비제품의 하나인 고밀도디스크수자음성체계가 도입되였다. CD 는 한면에 60 분이상의 많은 음성정보를 기억할수 있는 지울수 없는 디스크이다. CD 가 도입됨으로써 콤퓨터자료기억분야에서 혁신으로 되는 저가격빛디스크기억기술을 개발할수 있게 하였다. 여러가지 빛디스크체계를 표 5-3 에 소개하였다. 이것들을 간단히 고찰하자.

1. CD-ROM

음성 CD 와 CD-ROM(Compact Disk Read-Only Memory)은 모두 비슷한 기술을 가진다. 기본차이는 CD-ROM 구동기가 더 견고하며 자료가 디스크로부터 콤퓨터로 정확히 전송되도록 오유수정장치를 가지고 있다는것이다. 두 형태의 디스크들은 다 같은 방법으로 만든다. 디스크는 폴리카보네트와 같은 수지로 이루어 져 있으며 보통 알루미니움과 같은 센 반사면으로 씌워 져 있다. 수자적으로 기록된 정보(음악이나 콤퓨터자료)는 반사면우에 현미경적인 작은 구멍들로 찍히운다. 이것은 우선 디스크본판을 만들기 위해 높은 출력의 레이자로 초점을 정밀하게 맞추는 방법으로 진행된다. 그다음 디스크본판은 복제물로 찍어 내기 위한 형타를 만드는데 리용된다. 복제물의 구멍이 찍힌 표면은 맑은 락카로 피복하여 먼지와 긁히우는것을 막는다.

표 5-3. 빛디스크제품

CD
고밀도디스크. 수자식음성정보를 기억하는 지울수 없는 디스크이다. 표준체계는 12cm 디스크를 사용하며 련속적인 동작으로 60 분이상 되는 자료를 기록할수 있다.

CD-ROM
읽기전용기억기인 고밀도디스크. 콤퓨터자료를 기억하는데 리용하는 지울수 없는 기억기. 표준체계는 12cm 디스크를 사용하며 600Mbyte 이상 기록할수 있다.

DVD
수자식비데오디스크. 제작기술은 용량이 큰 수자식자료뿐아니라 압축된 비데오정보를 수자화한다.

WORM
한번 쓰고 여러번 읽기디스크. CD-ROM 보다 더 쉽게 씌여 지는 디스크. 한번의 복사를 하여 상품화할수 있다. CD-ROM 과 마찬가지로 쓰기조작이 진행된후에 디스크는 오직 읽을수만 있다. 가장 일반적인 크기는 5.25inch 인데 200Mbyte~800Mbyte 까지의 자료를 기록할수 있다.

Erasable Optical Disk
이 디스크는 빛기술을 리용하였지만 쉽게 지우고 다시 쓰기할수 있다. 3.25inch 와 5..25inch 디스크가 쓰인다. 일반적으로 용량은 650Mbyte 이다.

Magneto Optical Disk
이 디스크는 읽기 위한 빛기술과 빛초점을 조절하는 자기적기록기술을 리용하였다.
3.25inch 와 5.25inch 디스크가 쓰인다. 용량은 흔히 1Gbyte 이상이다.

정보는 빛디스크구동장치안에서 쪼여 지는 작은 출력의 레이자에 의하여 CD나 CD-ROM 으로부터 읽히워 진다. 레이자는 전동기가 디스크를 돌리여 레이자빛을 받게 하였을 때 맑은 보호표면을 거쳐 쪼여 진다. 구멍부분에서 반사되는 레이자빛의 세기는 다르다. 수감기는 이 변화를 수감하며 수자신호로 변환한다.

회전하는 디스크우의 중심가까이 있는 구멍은 바깥쪽의 구멍보다 더 천천히 고정된 점(레이자빛과 같은)을 지나며 따라서 레이자가 모든 구멍들을 같은 속도로 읽을수 있도록 속도의 변동을 보상할수 있는 몇가지 방법이 있다. 이것은 자기디스크와 같이 디스크 토막들에 기록된 정보비트사이의 공간을 늘임으로써 실현할수 있다. **등각속도(CAV)**의 고정된 속도로 디스크를 회전시킴으로써 정보를 같은 속도로 읽을수 있다. 그림 5-8 ㄱ에 CAV를 리용한 디스크의 설계를 보여 주었다. 디스크는 많은 분구들로 나누어 지며 동심원의 자리길들로 나누어 진다. CAV를 리용하는 우점은 개별적인 자료블로크들이 자리길과 분구들로 직접 주소화될수 있다는것이다. 현재위치로부터 지적된 주소로 자두를 이동하기 위해 지적된 자리길에로의 자두의 순간이동과 자두아래에서 돌아 가는 지적된 분구를 위한 짧은 대기가 있다. CAV 의 결함은 긴 바깥자리길에 보관될수 있는 자료의 량이 짧은 안쪽자리길에 보관될수 있는 자료의 량과 같은것이다.

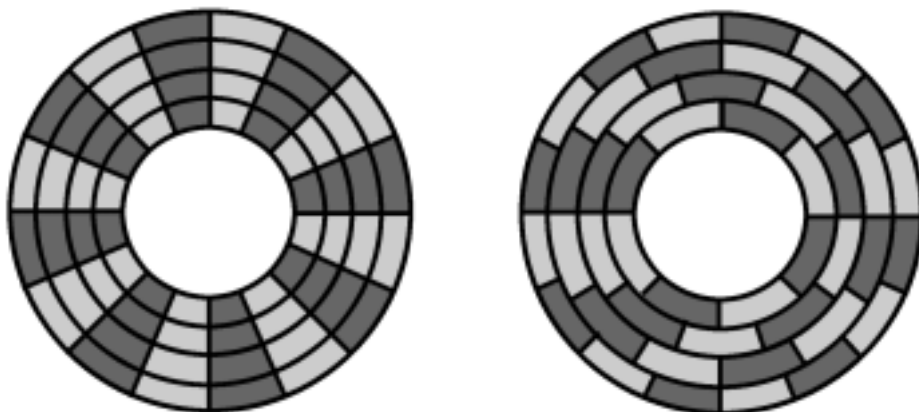



그림 5-8. 디스크편성방법의 비교
ㄱ-등각속도(CAV) ㄴ-등속선형속도(CLV)

CAV 방법은 디스크의 바깥쪽에는 적은 량의 정보밖에 기억할수 없으므로 많은 공간을 랑비하기때문에 CD와 CD-ROM에 리용되지 않는다. 반대로 정보를 같은 크기의 토막들로 묶으면 이것들은 부등속도로 회전하는 디스크에 의해 같은 속도로 주사된다. 구멍들은 일정한 **선속도(CLV)**로 레이자에 의하여 읽어 진다. 디스크는 중심으로부터 멀리 있는 호출일수록 더 천천히 회전한다. 이렇게 자리길의 용량과 회전지연은 바깥쪽으로 가면서 증가한다.

여러가지 밀도를 가진 CD-ROM들이 생산되고 있다. 표준디스크의 자리길과 자리길 사이의 공간은 1.6㎛(1.6×10^{-6}m)이다. CD-ROM의 기록할수 있는 자리길의 수는 32.55㎜의 반경을 자리길공간으로 나누어 보면 알수 있는데 결국 20344 개의 자리길이 있는것으로 된다. 사실상 하나의 라선형자리길이 있는데 이것은 라선의 회전수로 평균원의 둘레를 곱하는 방법으로 그 자리길의 길이를 구할수 있으며 이것은 약 5.27 ㎞로 계산된다.

CD-ROM 의 주어 진 선속도는 1.2 ㎧이므로 자리길을 한번 지나가는에 총 4391s 즉 73.2min 걸리며 이것은 음성 CD 에 표준적인 최대동작시간과 비슷하다. 자료가 176.4kbyte/s 의 속도로 디스크로부터 흐르기때문에 CD-ROM 의 기억용량은 774.5Mbyte 이다. 이것은 550 개의 3.25inch 플로피디스크의 용량과 맞먹는다.

CD-ROM 의 자료는 련속적인 블로크로 이루어 진다. 표준블로크형식을 그림 5-9 에 보여 주었다. 이것은 다음의 마당으로 이루어 진다.

- **동기:** 동기마당은 블로크의 시작을 식별한다. 그것은 모두 0 으로 된 1byte 와 모두 1 로 된 10byte 그리고 모두 0 으로 된 1byte 로 구성되여 있다.
- **머리부:** 머리부는 블로크주소와 방식바이트를 가진다. 방식 0 은 공백자료마당을 지정한다. 방식 1 은 오유수정부호와 2048byte 의 자료의 사용을 지정하며 방식 2 는 오유수정부호가 없는 2336byte 의 사용자자료를 지정한다.
- **자료:** 사용자자료
- **보조량:** 방식 2 에 따르는 추가적인 사용자자료이다. 방식 1 에서 이것은 288byte 의 오유수정부호이다.

그림 5-8 ㄴ는 CD 와 CD-ROM 에 사용된 등선속도방식의 배치를 보여 준다. CLV 사용으로 자유호출이 더 힘들어 진다. 지적된 주소를 알아 내자면 해당한 범위로 자두를 이동시키고 회전속도를 조절하며 주소를 읽은 다음 지적된 분구를 찾고 호출하기 위한 세부적인 조종을 진행하여야 한다.

CD-ROM 은 많은 사용자들이 많은 량의 자료를 서술하게 하는데 합리적이다. 초기 쓰기공정에서 많은 비용을 요구하기때문에 개별적인 응용에는 적합지 않다. 전통적인 자기디스크와 비교해 보면 CD-ROM 은 다음의 3 가지 우점을 가진다.

- 정보기억용량은 빛디스크가 훨씬 더 크다.
- 빛디스크는 자기디스크와 달리 거기에 기억된 정보들을 값 눅게 다량 재현할수 있다. 자기디스크상의 자료기지는 두 디스크구동기를 리용하여 한번에 하나의 디스크복사하는 방법으로 새끼친다.
- 빛디스크는 넣었다꺼냈다할수 있기때문에 많은 량의 정보자료를 디스크채로 축적해 두는데 리용할수 있다. 대부분의 자기디스크는 넣었다꺼냈다할수 없다. 넣었다꺼냈다할수 없는 자기디스크상의 정보는 디스크구동기나 디스크가 새 정보를 보관하는데 리용할수 있도록 하기에 앞서 우선 테프에 복사하여야 한다.

CD-ROM 의 결함은 다음과 같다.

- 읽기만 하고 갱신할수 없다.
- 자기디스크구동기보다 0.5s 정도의 더 긴 호출시간을 가진다.

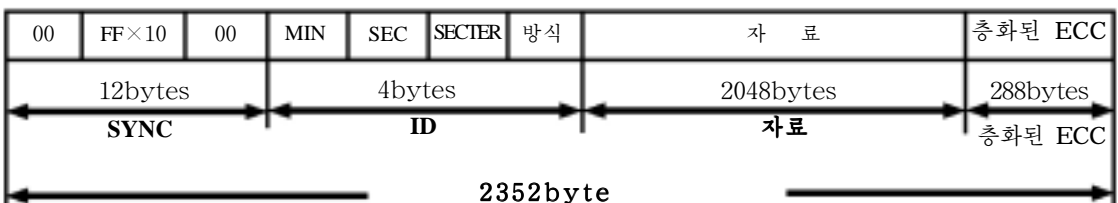



그림 5-9. CD-ROM 블로크형식

2. WORM

자료묶음에 대한 한번 혹은 여러번의 복사를 요구하는 응용을 위하여 한번의 쓰기와 여러번의 읽기를 진행하는 CD가 개발되였다. WORM(Write-Once Read-Many)에서 디스크는 적당한 세기의 레이자빛으로 한번 더 쓰기할수 있다. 그러므로 CD-ROM 보다 더 비싼 디스크조종기를 가지고 사용자는 읽기와 마찬가지로 한번은 쓰기할수 있다. 더 빠른 호출을 얻기 위하여 WORM은 일부 용량이 손실되는 CAV 방식을 리용한다.

디스크를 만드는 표준기술은 디스크우에 기포들을 형성시키는데 높은 출력의 레이자를 리용하는것이다. 미리 형식화된 매체가 WORM 구동기에 자리잡았을 때 작은 출력의 레이자가 충분한 열을 내여 이미 기록된 기포들을 터칠수 있다. 디스크읽기조작이 진행될 때에 WORM 구동기에서는 레이자가 디스크표면을 비친다. 터진 기포들이 그 주변과 높은 대조를 이루기때문에 이것들이 간단한 전자장치에 의하여 쉽게 인식된다.

WORM 빛디스크는 문서나 파일을 기억시키는데 많이 쓴다. 이것은 많은 량의 사용자자료를 영구보관하는데도 쓰인다.

3. 지울수 있는 빛디스크

지울수 있는 빛디스크는 임의의 자기디스크와 같이 반복적으로 쓰기와 덧쓰기를 할수 있다. 비록 많은 연구가 진행되였지만 순수 광학적방법(앞에서 론의된 자기광학적방법과 대치되는)만을 상변환방법이라고 한다. 상변환디스크는 두개의 각이한 상태에서 크게 차이나는 두가지 반사능을 가지는 재료를 리용한다. 여기에는 분자들이 무질서하게 운동하며 빛을 잘 반사하지 않는 무정형상태 그리고 빛을 잘 반사하는 매끈한 표면을 가진 결정상태가 있다. 레이자빛묶음은 재료를 한상으로부터 다른 상으로 변화시킬수 있다. 상변환빛디스크의 본질적인 결함은 재료들이 자기의 요구되는 성질을 때때로 영구적으로 잃어 버린다는것이다. 현재 리용하고 있는 재료들은 500,000~1,000,000회 지울수 있다.

지울수 있는 빛디스크는 CD-ROM과 재쓰기할수 있는 WORM에 비하여 고유한 우점을 가지며 2차기억기로 리용된다. 그러므로 이것은 자기디스크와 비슷하다. 자기디스크에 비한 지울수 있는 빛디스크의 본질적인 우점은 다음과 같다.

- **큰 용량**: 5.25inch 빛디스크는 약 650Mbyte의 자료를 기억할수 있다. 대부분 현대적인 윈체스터디스크는 이 용량의 절반이하이다.
- **이동성**: 빛디스크는 구동기에서 떼낼수 있다.
- **믿음성**: 빛디스크에 대한 공학적인 허용오차는 큰 용량의 자기디스크보다 훨씬 작다. 그러므로 그것들은 더 높은 믿음성과 더 긴 수명을 보장한다.

지울수 있는 빛디스크에서는 WORM에서처럼 CAV 방법을 리용한다.

4. 수자식비데오디스크

용량이 큰 수자식비데오디스크(DVD)가 개발됨으로써 전자공업부분에서는 드디여 상사식 VHS 비데오테프를 교체할수 있게 되였다. DVD는 비데오카세트록화기(VCR)에 사용된 비데오테프대신에 리용할수 있으며 여기서 더 중요한것은 이것을 개인용콤퓨터와

봉사기에서 CD-ROM 대신에 리용할수 있다는것이다. DVD 는 수자시대의 비데오이다. 그것은 화상질이 높은 영화를 보존하며 DVD 구동기에서도 동작할수 있는 음성 CD 와 같이 자유로 호출할수 있다. CD-ROM 보다 실지로 7 배에 달하는 방대한 량의 자료를 기록할수 있다. DVD 의 방대한 기억용량과 선명한 질로 개인용콤퓨터에 의한 오락은 더 생동하게 될것이며 교육프로그람은 더 구체적인 비데오로 형상될것이다. 이 발전으로 하여 Web 싸이트에 소개되는 실물이 아닌 제품들로 인터네트와 집합인트라네트를 통한 상품거래가 진행되는 새로운 양상이 이루어 졌다.

여기에 CD-ROM 과 DVD 가 구별되는 가장 중요한 점이 있다.

- 표준 DVD 는 층마다 4.7Gbyte 를 기억하며 한방향쌍층 DVD 는 8.5Gbyte 를 기억한다.
- DVD 는 높은 질의 완전화면그림에 대하여 MPEG 라고 하는 비데오압축형식을 사용한다.
- 단일층 DVD 는 2 시간 30 분짜리 영화를 기억할수 있고 한편 쌍층 DVD 는 4 시간 이상되는 영화를 기억할수 있다.

5. 자기빛디스크

자기빛(MO)디스크구동기는 일반적으로 쓰이는 자기디스크체계의 능력을 높이는데 빛레이자를 사용한다. 기록기술은 근본적으로 자기적이다. 그러나 빛레이자를 자기기록머리부의 초점을 맞추는데 효과적으로 리용함으로써 더 큰 가능성을 얻게 되였다. 여기서 디스크는 높은 온도에서만 극성이 변화되는 물질로 피복되여 있다. 정보는 그 표면우에 있는 미세한 점을 가열하는데 레이자를 리용하고 다음 자기마당을 가함으로써 디스크우에 씌여 진다. 그 점이 식으면서 자기마당의 북남량극을 취하게 된다. 분극공정이 디스크우에서 물리적변화를 일으키지 않기때문에 처리공정은 몇번 반복할수 있다.

읽는 작용은 순수 빛으로 진행한다. 자력선의 방향은 분극된 레이자빛(쓰기조작에서보다 더 적은 출력)에 의해 검출된다. 특수한 점에서 반사되는 분극된 빛은 자기마당방향에 의존하는 그의 회전정도를 변화시킬것이다.

순수한 빛 CD 구동기에 비한 MO 구동기의 기본적인 우점은 디스크의 수명이다. 빛디스크에 대한 자료의 반복적인 다시쓰기는 매체를 점차적으로 파괴시킨다. MO 구동기는 이런 파괴가 없으므로 반복적인 다시쓰기를 계속할수 있다. MO 기술의 다른 우점은 그것이 자기기억기보다 Mbyte 당의 값이 아주 눅은것이다.

제 4 절. 자기테프

테프체계에서는 디스크체계에서와 같은 읽기와 기록기술을 리용한다. 매체는 자성산화물로 덮인 녹신녹신한 마일라테프이다. 테프와 테프구동기는 가정용록음기테프기록체계와 류사하다.

테프매체는 작은 수의 평행자리길들로 구성되여 있다. 초기에 나온 테프체계는 표준적으로 9 개 자리길을 사용했다. 이것은 한번에 1byte 의 자료를 기억할수 있으며 9 번째

자리길로서 보충적인 기우성비트를 가지고 있다. 더 새로운 테프체계에서는 하나의 수자단어 혹은 배단어에 해당하는 18 혹은 36 자리길을 사용한다. 디스크에서처럼 자료는 테프우에서 **물리적레부호**라고 하는 린접블로크에서 읽기되고 쓰기된다. 테프우의 블로크들은 기록사이 간극에 의하여 갈라 져 있다. 그림 5-10 은 9 개의 자리길을 가진 테프의 구조를 보여 준다. 디스크에서처럼 테프는 자리잡은 물리적레코드를 알아 내는데 도움이 되도록 형식화되여 있다.

테프구동기는 **순차호출장치**이다. 만일 테프머리부가 레코드 1 에 위치하면 그다음 레코드 N 을 읽기 위해서는 N-1 을 통해 단번에 하나씩 물리적레코드1 을 읽어야 한다. 만일 머리부가 요구되는 레코드경계밖에 위치하고 있다면 일정한 길이만큼 테프를 다시 감고 앞으로 읽어 나가기 시작하여야 한다. 디스크와 달리 테프는 오직 읽거나 쓰

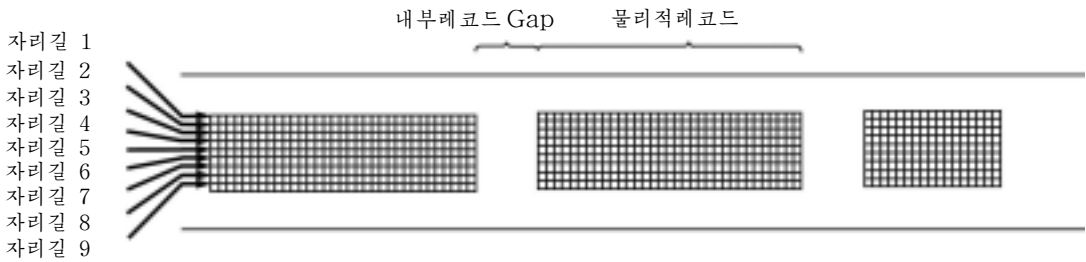



그림 5-10. 9 개 자리길의 자기디스크형식

기조작하는 동안만 이동상태에 있다.

테프와는 대조적으로 디스크구동기는 **직접호출장치**라고 한다. 디스크구동기는 요구하는 자료를 얻기 위하여 디스크우의 모든 분구를 차례로 읽을 필요가 없다. 그것은 오직 한 자리길안의 분구에 끼여 들어 가는것만 대기하여야 하며 임의의 자리길에 대한 련속 호출을 할수 있다.

자기테프는 보조기억기의 첫 종류였다. 자기테프는 여전히 기억기계층에서 값이 가장 눅고 속도가 가장 느린것으로서 널리 쓰이고 있다.

참고문헌과 Web 싸이트

[MEE96a]은 디스크와 테프체계의 레코드화기술을 기본적으로 개괄하였다. [MEE96b]는 디스크와 테프체계에 대한 자료기억기술을 기본으로 취급하였다.

RAID 개념의 발명가들에 의하여 작성된 RAID 기술의 우수한 개괄은 [CHEN94]에서 주었다. 더 자세한 론의는 RAID제품에 대한 공급자와 구입자들의 련합인 RAID자문기구에 의하여 공개되였다[MASS97]. 최근 가장 좋은 연구론문은 [FRIE96]이다.

[MARC90]은 빛기억분야의 좋은 고찰을 주었다. 기본적인 기록과 읽기기술에 대한 개괄은 [MANS97]에서 주었다.

마지막으로 [ROSC97]은 개별적인 장치들에 대한 상세한 기술의 전부로서 외부기억체계의 모든 형태를 폭 넓게 고찰하였다.

CHEN94 Chen, p.;Lee, E.; Gibson, G.; Katz, R.;and Patterson, D. "RAID: High Performance, Reliable Secondary Storage." *ACM Computing Surveys*, June 1994.
FRIE96 Friedman, M. "RAID Keeps Going and Going and…" *IEEE Spectrum*, April 1996.
MANS97 Mansuripur, M.,and Sincerbox, G."Principles and Techniques of Optical Data Storage." *Proceedings of the IEEE*, November 1997.
MARC90 Marchant, A. *Optical Recording*. Reading, MA:Addison-Wesley, 1990.
MASS97 Massiglia, P. *The RAID Book: A Storage System Technology Handbook*. St. Peter, MN: The Raid Advisory Board, 1997.
MEE96a Mee, C., and Daniel, E., eds. *Magnetic Recording Technology*. New York: McGraw-Hill, 1996.
MEE96b Mee, C.,and Damiel, E., eds. *Magnetic Storage Handbook*. New York: McGraw-Hill, 1996.
ROSC97 Rosch, W. *Winn L.. Rosch Hardware Bible*. Indianapolis, IN: Sams, 1997.

Web 싸이트:

- RAID 자문기구: RAID 산업그룹

련습문제

1. 디스크체계에 대하여 다음것을 정의하시오.

   t_s – 탐색시간; 자리길우의 자두가 자리를 잡는데 걸리는 평균시간
   r – 디스크의 회전속도; 초당 회전수
   n – 분구당 비트수
   N – 자리길의 용량; 비트
   t_A – 분구호출시간
   다른 지표들의 함수로서의 t_A에 관한 식을 전개하시오.

2. 10-구동기 RAID 의 구성을 가정하자. 다음란들을 채우고 여러가지 RAID 준위들을 비교하시오.

RAID 준위	기억밀도	대역너비성능	취급성능
0	1		1
1			
2			
3		1	
4			
5			

매 지표들은 가장 좋은 성능을 내는 RAID 준위로 표준화되였다. 그러므로 란에 남아 있는 수들은 0과 1사이의 값을 가져야 한다. 기억기의 밀도란 사용자자료를 디스크기억기의 능력으로 나눈 값을 말한다. 띠너비성능은 자료가 얼마나 빠른 속도로 배렬밖으로 전송될수 있는가를 반영한다. 처리성능은 한 배렬이 초당 얼마나 많은 I/O 조작을 수행할수 있는가 하는 척도이다.

3. 디스크줄짓기는 줄크기가 I/O 요구크기에 비하여 작을 때에 자료전송속도를 높일수 있다는것을 설명하시오. 또한 RAID 0은 다중 I/O 요구는 병렬로 조종될수 있기때문에 단순한 큰 디스크에 비하여 높은 성능을 보장한다는것을 설명하시오. 그러나 이 두번째 문제에서 디스크줄짓기가 필요하겠는가? 즉 디스크줄짓기가 줄짓기가 안된 대비디스크배렬에 비하여 I/O 속도성능을 높이겠는가?

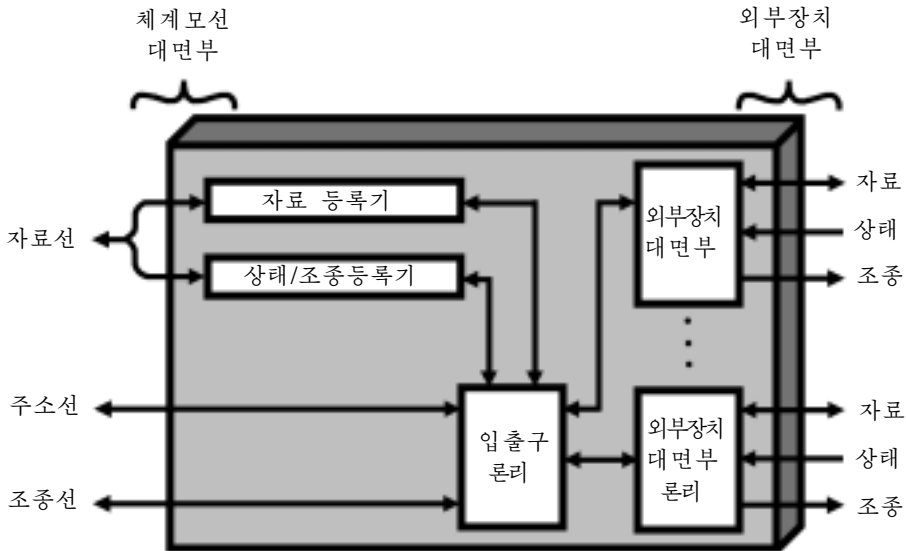
4. 테프속도가 초당 120inch 이고 테프밀도가 인치당 1600bit 인 자기테프장치의 9자리길자성테프의 전송속도는 얼마인가?

5. 2400ft(1ft=30cm)의 테프 한틀이 있다고 하자.
레코드사이간극은 0.6inch, 테프의 속도는 간극이 선형적일 때 증가 또는 감소한다. 그리고 테프의 다른 특성들은 문제 4번에서와 같다. 테프우의 자료들은 물리적레코드로 구성되여 있으며 매 물리적레코드는 론리적레코드라고 하는 고정적인 사용자정의단위수를 포함하고 있다.

   ㄱ. 물리적레코드 하나당 10개씩 블로크화된 120byte 론리적레코드의 전체 테프를 읽는데 얼마만한 시간이 걸리겠는가?
   ㄴ. 물리적레코드 하나당 30개씩 블로크화된 120byte 론리적레코드의 전체 테프를 읽는데 얼마만한 시간이 걸리겠는가?
   ㄷ. 앞에서 말한 블로크화인자들을 가진 테프가 얼마나 많은 론리적레코드들을 유지하고 있겠는가?
   ㄹ. 앞에서 말한 2개의 블로크화인자들에 관하여 총 유효전송속도는 얼마이겠는가?
   ㅁ. 테프의 용량은 얼마인가?

6. 만일 디스크가 512byte/분구, 96분구/자리길, 110자리길/면의 고정분구식이고 8개의 사용가능한 면을 가지고 있다면 문제 5번의 ㄴ에서 읽은 론리적레코드들을 기억하는데 디스크공간(분구, 자리길, 면)이 얼마나 요구되는가를 계산하시오. 임의의 파일머리부레코드와 자리길첨수를 무시하고 레코드들은 한 분구이상 회전할수 없다고 가정하시오.

제 6 장. 입출력

◆◆◆

- ◆ 콤퓨터체계의 I/O 기본방식은 외부세계와의 대면부이다. 이 기본방식은 외부세계와의 호상작용을 조종하는 체계수단을 제공하며 I/O 작용을 효과적으로 관리하는데 요구되는 정보를 조작체계에 제공하도록 설계된다.
- ◆ 다음과 같은 세가지 기본 I/O 기술이 있다. **프로그람식 I/O** 에서는 I/O 조작을 요구하는 프로그람의 직접 및 련속조종하에서 I/O 조작이 진행된다. **새치기구동식 I/O** 에서는 프로그람이 I/O 지령을 출력하고 I/O 조작의 끝을 알려 주는 I/O 하드웨어에 의하여 중단될 때까지 실행이 계속된다. **직접기억기접근(DMA)**은 전용 I/O 처리장치가 큰 자료블로크를 이동시키는 I/O 조작을 조종한다.
- ◆ 외부 I/O 대면부의 대표적인 실례에는 SCSI 와 FireWire 의 두가지가 있다. **SCSI** 는 외부장치에 대한 병렬대면부이며 한편 보다 새로운 **FireWire** 는 고속직렬대면부이다.

◆◆◆

처리장치와 기억기모듈묶음외에 콤퓨터체계의 세번째 주요요소는 I/O 모듈묶음이다. 매 장치는 체계모선이나 중심스위치들에 접속되여 하나이상의 주변장치들을 조종한다. I/O 모듈는 단순히 체계모선에 장치를 련결시키는 기계적련결기들의 묶음이 아니다. 오히려 I/O 모듈는 일련의 《지능》을 가진다. 즉 주변장치와 모선사이에서 통신기능을 수행

하기 위한 론리를 포함한다.

독자들은 왜 주변장치들을 직접 체계모선에 련결시키지 않는가 하고 이상하게 여길수 있다. 그 리유는 다음과 같다.

- 여러가지 조작방법을 가진 아주 다양한 주변장치들이 있다. 여러 장치들을 조종하기 위하여 처리장치내에 필요한 론리적장치를 설치하는것은 비현실적일것이다.
- 보통 주변장치들의 자료전송속도는 기억기나 처리장치의 자료전송속도보다 훨씬 더 느리다. 따라서 고속체계모선을 리용하여 직접 주변장치와 통신하는것은 비현실적이다.
- 주변장치들은 흔히 그것들이 붙어 있는 콤퓨터에서 보다 각이한 자료형식들과 단어길이들을 리용한다.

그리하여 I/O 모듈이 요구된다. 이 장치는 두가지 주요기능을 가진다(그림 6-1).

- 체계모선 또는 중앙스위치를 거쳐 처리장치나 기억기와 대면한다.
- 적당한 자료련결에 의하여 하나이상의 주변장치와 대면한다.

이 장에서는 외부장치들에 대한 간단한 론의로부터 시작하여 I/O 모듈의 구조와 기능을 고찰하였다. 그다음 처리장치와 기억기가 협동하여 I/O 기능을 수행할수 있는 여러가지 방법들 즉 내부 I/O 대면부를 고찰한다. 끝으로 I/O 모듈과 외부세계사이의 외부 I/O 대면부를 조사한다.

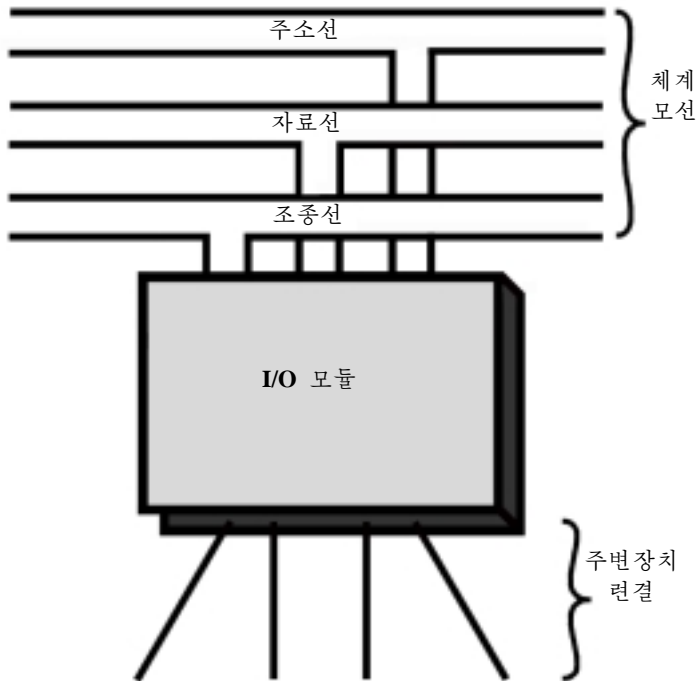



그림 6-1. I/O 모듈의 일반모형

제 1 절. 외부장치

I/O 조작들은 외부환경과 콤퓨터사이에서 자료를 교환하는 수단으로 되는 많은 외부장치들을 통하여 실현된다. 외부장치는 I/O 모듈에 련결됨으로써 콤퓨터에 접속된다(그림 6-1). 련결은 I/O 모듈과 외부장치사이에서 조종, 상태, 자료를 교환하는데 리용된다. I/O 모듈에 련결된 외부장치를 흔히 주변장치라고 한다.

주변장치는 크게 세가지로 분류할수 있다.

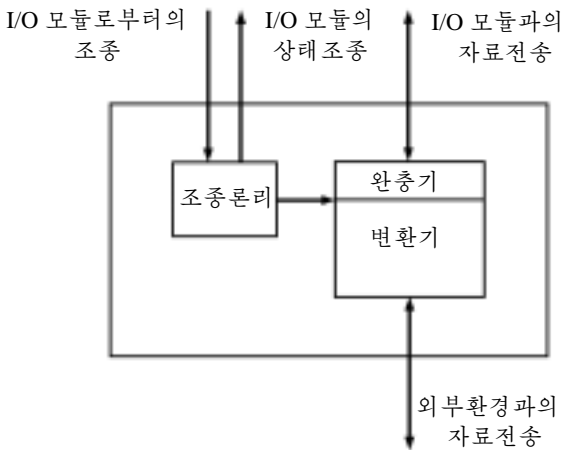
- **사람이 읽을수 있는 장치**(Human readable): 콤퓨터사용자와의 통신에 적합하다.
- **기계가 읽을수 있는 장치**(Machine readable): 장치와의 통신에 적합하다.
- **통신하는 장치**(Communication): 떨어 져 있는 장치들과의 통신에 적합하다.

사람이 읽을수 있는 장치의 실례는 현시말단(VDT)과 인쇄기이다. 기계가 읽을수 있는 장치의 실례는 자기원판과 테프체계, 로보트공학에 리용되는 수감부와 전동장치 등이다. 이 장에서는 I/O 모듈로서의 디스크와 테프를 고찰하며 제 5 장에서는 이것들을 기억장치로서 고찰하였다. 기능상 견지에서 보면 이 장치들은 기억기의 한 부분이며 그 리용에 대해서는 이미 제 5 장에서 대체로 론의되였다. 구조적견지에서 보면 이 장치들은 I/O 모듈에 의하여 조종되므로 이 장에서 고찰하려고 한다.

그림 6-2. 외부장치

통신장치들은 콤퓨터가 떨어 져 있는 장치와 자료를 교환할수 있게 한다. 떨어 져 있는 장치로는 말단과 같은 사람이 읽을수 있는 장치, 기계가 읽을수 있는 장치 또는 다른 콤퓨터가 될수 있다.

아주 일반적으로 표현하면 외부장치의 본질은 그림 6-2 에서 보여준바와 같다. I/O 모듈과의 대면부는 조종신호, 자료 및 상태신호의 형식으로 되여 있다. **조종신호**는 자료를 장치에 보내기(INPUT 또는 REAL), 자료를 I/O 모듈로부터 받기(OUTPUT 또는 WRITE), 상태보고 또는 장치에 대한 특수한 조종기능수행(즉 디스크자두를 지적)과 같은 그 장치가 수행하게 되는 기능을 결정한다. **자료**는 비트들의 묶음형태로 I/O 모듈에 보내 지거나 I/O 모듈로부터 받는다. **상태신호**는 장치의 상태를 지적한다. 장치가 자료전송준비가 되였는가를 보여 주는 READ/NOT-READY 가 그 실례로 된다.

장치와 관련한 **조종론리**는 I/O 모듈로부터의 방향에 따라 장치동작을 조종한다. **변환기**는 출력할 때에는 자료를 전기에네르기로부터 다른 형태의 에네르기로 변환하며 입력할 때에는 다른 형태의 에네르기를 전기에네르기로 변환시킨다. 일반적으로 완충기는 변

환기와 결합되여 I/O 모듈과 외부환경사이에서 전송되는 자료를 일시적으로 보관한다. 완충기크기는 대체로 8～16bit 이다.

I/O 모듈과 외부장치사이의 대면부는 제 7 절에서 고찰한다. 외부장치와 환경사이의 대면부는 이 책에서 취급하는 내용이 아니지만 여기서는 몇가지 간단한 실례들을 보여 준다.

1. 건반/감시장치

표 6-1. ASCII 비트자리

							0	0	0	0	1	1	1	1
							0	0	1	1	0	0	1	1
							0	1	0	1	0	1	0	1
b_7	b_6	b_5	b_4	b_3	b_2	b_1								
			0	0	0	0	NUL	DLE	SP	0	@	P	`	P
			0	0	0	1	DOH	DC1	!	1	A	Q	a	q
			0	0	1	0	STX	DC2	“	2	B	R	B	r
			0	0	1	1	ETX	DC3	#	3	C	S	c	s
			0	1	0	0	EOP	DC4	$	4	D	T	d	t
			0	1	0	1	ENQ	NAK	%	5	E	V	e	u
			0	1	1	0	ACK	SYN	&	6	F	W	f	v
			0	1	1	1	BEL	ETB	‘	7	G	X	g	w
			1	0	0	0	BS	CAN	(	8	H	Y	h	x
			1	0	0	1	HT	EM	)	9	I		i	y
			1	0	1	0	LF	SUB	*	:	J	Z	j	z
			1	0	1	1	VT	ESC	+	;	K	[	k	{
			1	1	0	0	FF	FS	,	<	L	＼	l	|
			1	1	0	1	CR	GS	-	=	M	]	m	}
			1	1	1	0	SO	RS	.	>	N	^	n	~
			1	1	1	1	SI	US	/	?	O	-	o	DEL

콤퓨터와 사용자가 호상작용하는 가장 공통적인 수단으로는 건반/감시장치이다. 사용자의 입구는 건반이다. 자료는 이 입구를 통하여 콤퓨터에 전송되며 또한 감시장치에도 표시된다. 그외에 감시장치는 콤퓨터가 제공하는 자료를 표시한다.

기본교환단위는 문자이다. 코드는 문자에 관계되며 대체로 7～8bit 의 길이를 가진다. 가장 일반적으로 리용되는 코드는 ASCII 라고 하는 7bit 코드이다. 이 코드에서 매 문자

는 단일한 7bit 2진부호로 표시된다. 따라서 128개의 각이한 문자들을 표시할수 있다. 표 6-1 은 모든 코드값들을 보여 준다. 표에서 매 문자의 비트들은 맨 웃자리비트(MSB)인 b_7로부터 맨 아래자리비트(LSB)인 b_1로 표시된다. 문자에는 두가지 형태 즉 인쇄가능문자와 조종문자가 있다[1](표 6-2). 인쇄가능문자는 종이에 인쇄할수 있거나 화면에 표시할수 있는 자모,수자,특수문자들이다. 례를 들어 문자"K"의 비트표시는 1001011이다. 일부 조종문자들은 문자들을 인쇄하거나 표시하는것을 조종할수 있어야 한다. 례를 들면 행바꾸기(CR:Carrige Return)조종문자이다. 다른 조종문자들은 통신수속(communications procedure)과 관계된다.

건반입력인 경우 사용자가 건을 누르면 이것은 건반에 있는 변환기에 의하여 해석되여 대응하는 ASCII코드의 비트패턴으로 변환되는 전기적신호를 발생한다. 그 다음 이 비트패턴을 콤퓨터안의 I/O 모듈로 전송한다. 콤퓨터에서 본문은 같은 ASCII코드로 기억될수 있다. 출력인 경우는 ASCII 문자들이 I/O 모듈로부터 외부장치에로 전송된다. 외부장치에서 변환기는 이 코드를 해석하여 출력장치에 련속적인 문자렬을 표시하거나 필요한 조종기능을 수행하도록 전기적신호를 보낸다.

2. 디스크구동기

디스크구동기에는 자료,조종,상태신호들을 I/O 모듈과 교환하기 위한 전자장치와 디스크 읽기/쓰기기구를 조종하는 전자장치들이 있다. 고정-머리디스크에서 변환기는 움직이는 디스크겉면에 대한 자기적패턴과 그 장치완충기에서 비트들사이를 변환시킬수 있다 (그림 6-2). 이동-머리디스크는 디스크팔이 디스크겉면에서 왔다갔다 움직일수 있게 되여 있다.

표 6-2. ASCII 조종문자

형식조종	
BS(Backspace): 인쇄기구의 움직임을 표시하거나 현시장치의 유표를 한자리 뒤로 이동시키는것을 가리킨다.	**VT**(Vertical tab): 인쇄기장치구나 현시장치유표를 미리 할당된 인쇄행의 다음계렬로 이동시킨다는 것을 가리킨다.
HT(Horizontal tab): 인쇄기구나 현시장치유표를 다음 미리할당 "**tab**" 혹은 정지위치쪽으로 이동시키는것을 가리킨다.	**FF**(Form feed) : 인쇄기구나 현시장치유표가 다음 페지,양식,화면의 첫 위치에 놓인다는것을 가리킨다.
LF(Line feed): 인쇄기구나 현시장치유표를 다음행의 시작위치로 이동시킨다는것을 지적한다.	**CR**(Carriage return): 인쇄기구나 현시장치유표를 같은 행의 시작위치에 놓는다.

[1] 출구에서 ASCII 코드문자는 I/O 모듈로부터 외부장치에로 전송된다. 장치에서 변환기는 이 코드를 번역하여 요구되는 전기적신호들을 출력장치로 보내여 지적된 문자를 표시하거나 요구되는 조종기능을 수행한다.

전송조종

SOH(Start of heading): 머리부의 시작을 지적하는데 리용된다. 머리부는 주소나 경로조종정보를 가질수 있다.

STX (Start of text): 본문의 시작을 지적하며 머리부의 끝을 지적하기도 한다.

ETX(End of text):**STX** 로부터 시작된 본문을 끝내는데 리용된다.

EOT (End of transmission): 머리부가 있는 하나이상의 "본문"의 전송끝을 지적한다.

ENQ(Enquiry): 원격국으로부터의 응답요구. 국이 그 자체를 식별하는것은 "당신은 누구인가"라는 요구처럼 리용될수도 있다.

ACK(Acknowledge): 송신자에로의 긍정응답으로서 수신장치가 전송하는 문자. 그것은 문의통보에 대한 답례응답으로서 리용된다.

NAK(Negative acknowledge): 송신자에로의 부정응답으로서 수신장치가 보내는 문자. 그것은 문의통보에 대한 답례응답으로서 리용된다.

SYN(Synchronous/idle): 동기를 보장하는 동기전송체계에 리용된다. 어떠한 자료도 보내여 지지 않을 때 동기전송체계는 **SYN** 문자를 련속 보낸다.

ETB(End of transmission block): 통신목적을 위하여 블로크자료의 끝을 지적한다. 블로크구조가 처리형식에 필수적으로 관계되지 않는 블로크자료에 리용된다.

정보분리기

FS (File separator)

GS (Group separator)

RS (Record separator)

US (United separator)

그것들의 계층이 **FS～US** 인것을 제외하면 선택방법으로 리용되는 정보분리기

기타

NUL(Null): 문자가 없다. 자료가 없을 때 시간채우기나 테프공간채우기에 리용된다.

BEL(Bell): 사람의 주의를 끌 필요가 있을 때 리용된다. 자명종이나 주의장치들을 조종할수 있다.

SO(Shift out): 뒤따르는 코드결합이 SI 문자가 나타날 때까지 비표준문자묶음의 형태로 해석된다는것을 가리킨다.

SI(Shift in): 뒤따르는 코드결합이 표준문자묶음의 형태로 해석된다는것을 가리킨다.

DEL(Delete): 요구되지 않는 문자들을 지우는데 리용된다. 례를 들어 덧쓰기

SP(Space): 단어들을 분리하거나 인쇄기구 또는 현시장치유표를 한 자리 앞으로 이동시키는데 리용되는 인쇄되지 않는 문자

DLE(Data link escape): 하나이상의 련속적인 다음문자들의 뜻을 변화시키는 문자. 이것은 보조적조종을 제공하거나 임의의 비트결합을 가지는 자료문자들의 송신을 허락한다.

DC1, **DC2**, **DC3**, **DC4**(Device controls): 보조장치나 특수한 말단특징들의 조종을 위한 문자

CAN(Cancel): 통보문이나 블로크에서 선행한 자료가 무시된다는것을 지적한다.

EM(End of medium): 테프나 다른 매체의 물리적 끝 또는 매체의 요구되거나 리용된 부분을 지적한다.

SUB(Substitute): 오유 또는 무효로 된 글자를 치환한다.

ESC(Escape): 규정된 수의 련속문자렬을 주는 코드확장문자

제 2 절. I/O 모듈

1. 모듈함수

I/O 모듈에 대한 주요기능 혹은 요구는 다음과 같이 분류한다.

- 조종과 시간일치(동기화)
- 처리장치통신
- 장치통신
- 자료완충
- 오유검사

어떤 시간동안에 처리장치는 I/O에 대한 프로그람들의 요구에 따라 예언할수 없는 형식으로 하나이상의 외부장치들과 통신할수 있다. 주기억기와 체계모선과 같은 내부자원들은 자료 I/O를 포함하여 많은 동작들에서 공유되여야 한다. 그리하여 I/O 기능은 **조종과 시간일치**요구를 포함하여 내부자원들과 외부장치들사이에서 통신량의 넘침을 조절한다. 례를 들어 외부장치로부터 처리장치에로의 자료전송에 대한 조종은 다음과 같은 순서로 진행된다.

- 처리장치는 I/O 모듈에 접속된 장치의 상태를 검사할것을 요구한다.
- I/O 모듈는 장치상태를 처리장치에 보고한다.
- 장치가 동작하여 전송준비가 되였다면 처리장치는 I/O 모듈에 하나의 지령을 보내여 자료전송을 요구한다.
- I/O 모듈은 외부장치로부터 한 단위의 자료(즉 8∼16bit)를 얻는다.
- 자료는 I/O 모듈로부터 처리장치로 전송된다.

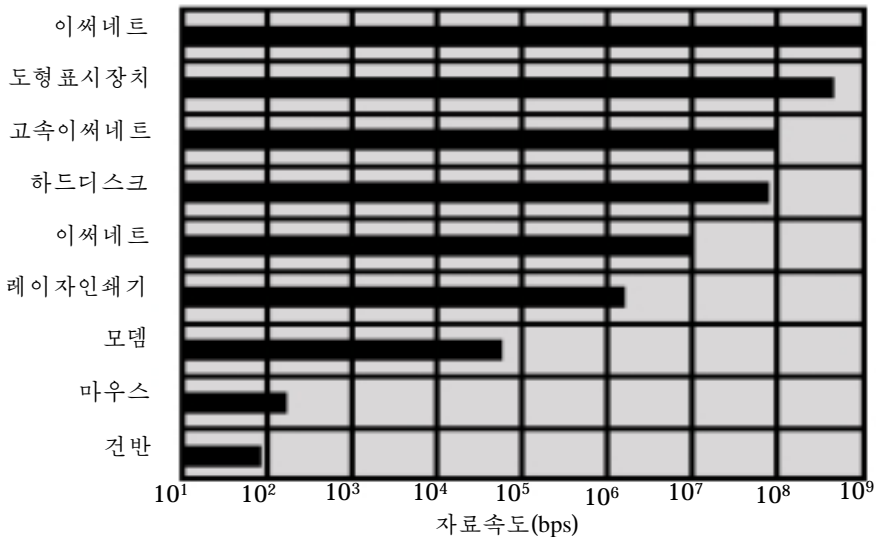



그림 6-3. 대표적 I/O 모듈의 자료속도

만일 체계가 모선을 리용한다면 모든 처리장치와 I/O 모듈사이 호상작용은 하나이상의 모선중재를 동반한다.

선행한 단순한 조작은 I/O 모듈이 처리소자와 통신해야 하며 또한 외부장치와도 통신해야 한다는것을 례증한다. **처리소자통신**은 다음과 같은것을 포함한다.

- **지령해신:** I/O 모듈은 처리소자로부터 조종모선에 신호로 보내진 지령들을 접수한다. 례를 들어 디스크구동기에 대한 I/O 모듈는 다음과 같은 지령 즉 READ SECTOR, WRITE SECTOR, SEEK 트랙크수, SCAN 레코드 ID 를 접수한다. 다음의 두 지령들은 다 자료모선에 보내지는 파라메터를 포함한다.
- **자료:** 자료는 자료모선에 의하여 처리소자와 I/O 모듈사이에서 교환된다.
- **상태보고:** 주변장치가 너무 느리기때문에 I/O 모듈의 상태를 아는것이 중요하다. 례를 들어 처리장치가 I/O 모듈에 자료를 보내겠는가고 묻는 경우 I/O 모듈은 여전히 그 전의 I/O 지령을 수행하고 있으므로 준비가 되지 않은 상태에 있다. 이러한 사실을 상태신호로 알릴수 있다. 공통적인 상태신호들은 BUSY 와 READY 이다. 여러가지 오유조건들을 보여 주는 신호들도 있다.
- **주소인식:** 기억기에 있는 매 단어가 주소를 가지는것처럼 I/O 모듈도 주소를 가진다. 그리하여 I/O 모듈은 그것이 조종하는 매 주변장치에 대한 유일한 주소를 인식해야 한다.

다른 한편 I/O 모듈은 **장치통신**을 수행할수 있어야 한다. 이 통신은 지령, 상태정보, 자료를 포함한다(그림 6-2).

I/O 모듈의 기본과제는 **자료완충**이다. 이 기능의 필요성은 그림 6-3 에서 명백히 보여 준다. 주기억기나 처리장치안으로 혹은 주기억기나 처리장치밖으로의 전송속도가 아주 높지만 그 속도는 대부분의 주변장치들에서 보다 낮으며 넓은 범위에 놓인다. 주기억기로부터 오는 자료는 매우 빨리 I/O 모듈로 전송된다. 자료는 I/O 모듈에 완충되며 그 다음 I/O 모듈의 자료속도로 주변장치에 전송된다. 자료전송이 반대방향으로 진행될 때에는 I/O 모듈의 느린 전송조작으로 기억기가 구속을 받지 않도록 자료를 완충한다. 결국 I/O 모듈은 두개의 장치 및 두 기억기속도로 동작할수 있어야 한다.

끝으로 I/O 모듈은 보통 **오유검사**와 처리장치에 대한 보조적인 오유통지를 할수 있다. 오유종류에는 장치에 의하여 보고되는 기계적 및 전기적고장(즉 종이고장,나쁜 디스크자리길)들이 포함된다. 또한 자료가 장치로부터 I/O 모듈로 전송되는것으로 인한 비트패턴에서의 비고의적인 변화들에 의하여 생기는 오유도 있다. 전송오유를 검사하는데는 흔히 일부 형태의 오유검사부호를 리용한다. 자료의 매 문자에 대하여 기우성비트를 리용하는것은 일반적인것으로 되고 있다. 례를 들어 ASCII 문자코드는 한 바이트중에서 7 개의 비트를 차지한다. 8번째 비트는 그 바이트에서 1의 전체수가 기수(기수기우성) 또는 우수(우수기우성)가 되도록 설정된다. I/O 모듈은 한 바이트가 수신되면 이 기우성비

트를 검사하여 오유가 발생했는가를 결정한다.

2. I/O 모듈구조

I/O 모듈에는 복잡성과 조종하는 외부장치의 수에 따라 여러가지가 있다. 여기서는 이에 대하여 매우 일반적으로만 서술한다(한가지 특별한 장치인 Intel 82C55A 에 대하여서는 제 4 절에서 고찰한다). 그림 6-4 는 I/O 모듈의 일반적인 구성도를 보여 준다. 장치는 조종선묶음(즉 체계모선)을 통하여 콤퓨터의 나머지부분들과 련결된다. 장치에로 그리고 장치로부터 전송되는 자료는 하나이상의 자료등록기들에 완충된다. 또한 현재 상태정보를 제공하는 하나이상의 상태등록기들이 있을수 있다. 상태등록기는 조종등록기로서 작용하므로 처리장치로부터 구체적인 조종정보를 받는다. 장치내에서의 론리는 조종선묶음을 통하여 처리장치와 호상작용한다. 조종선묶음은 처리장치가 I/O 모듈에 지령들을 보낼 때 리용된다. 일부 조종선 즉 중재와 상태신호선들은 I/O 모듈에 의하여 리용될수 있다. 장치는 또한 그것이 조종하는 장치와 관련한 주소들을 인식하고 발생시킬수 있어야 한다. 매 I/O 모듈은 유일한 주소를 가지며 하나이상의 외부장치를 조종하는 경우에는 유일한 주소묶음을 가진다. 끝으로 I/O 모듈은 그것이 조종하는 매 장치와의 대면부에 대하여 특별한 론리를 가지고 있다.

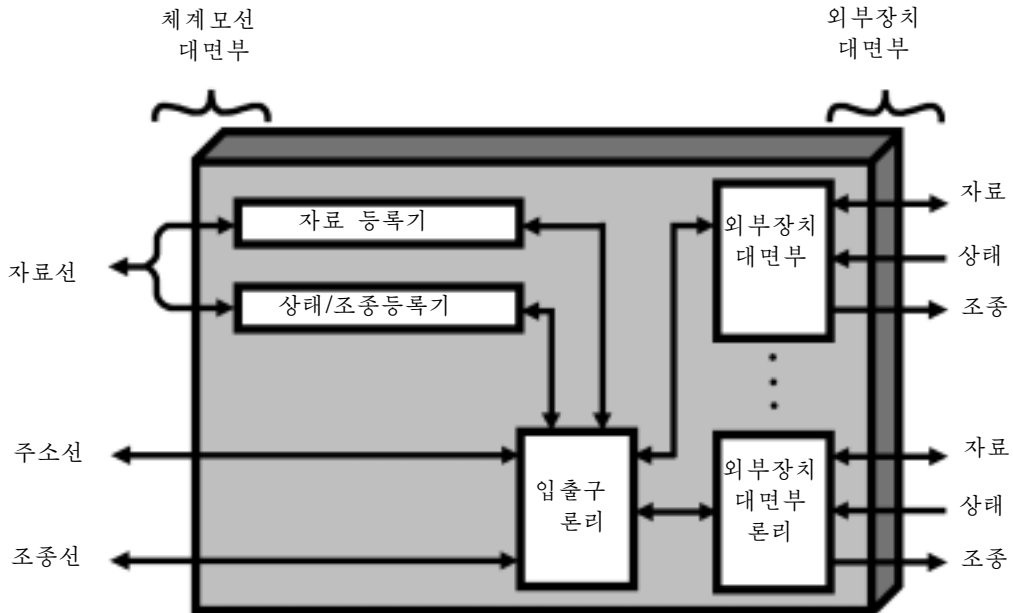



그림 6-4. I/O 모듈의 구성도

I/O 모듈은 처리장치가 단순한 방법으로 여러가지 장치들을 관리할수 있는 많은 능력들이 제공되여 있다. I/O 모듈은 시간일치,형식,외부장치의 전기기계부분 등의 상세한

내용을 은페시켜 처리장치가 간단한 읽기 및 쓰기지령,가능한 열기 및 닫기파일지령에 대한 기능을 수행할수 있게 한다. 가장 간단한 형태에서는 I/O 모듈이 여전히 처리장치에 있는 장치(레로 테프풀기장치)를 조종하기 위한 많은 작업량이 남아 있을수 있다.

처리장치와의 높은준위대면부를 실현하면서 상세한 처리부담의 대부분을 걸머지는 I/O 모듈은 보통 **I/O 통로** 혹은 **I/O 처리장치**이다. 아주 원시적인 그리고 세부적인 조종을 요구하는 I/O 모듈에는 **I/O 조종기**나 **장치조종기**가 있다. I/O 조종기는 일반적으로 극소형 콤퓨터에서 리용되며 I/O 통로는 대형콤퓨터에서 리용된다.

이제부터 혼란이 없을 때에는 I/O 모듈이라는 일반용어를 리용하며 필요할 때마다 보다 명백한 용어를 리용하겠다.

제 3 절. 프로그람식 I/O

I/O 조작에는 세가지 기술이 있다. **프로그람식** I/O 를 리용하여 자료를 처리장치와 I/O 모듈사이에서 교환할수 있다. 처리장치는 장치상태를 수감하고 읽기나 쓰기지령을 보내며 자료를 전송하는것을 포함하여 I/O 모듈를 직접 조종하는 프로그람을 실행한다. 처리장치는 I/O 모듈에 지령을 보낼 때 I/O 조작이 완성될 때까지 기다려야 한다. 처리장치가 I/O 모듈보다 속도가 더 빠른 경우 이 시간은 불필요한 처리장치시간이다. **새치기구동식** I/O 를 리용하여 처리장치는 I/O 지령을 내보내며 다른 명령들을 계속 집행하며 이 명령들이 완료되면 I/O 모듈에 의하여 새치기된다. 프로그람식 I/O 와 새치기구동 I/O 인 경우 처리장치는 출력할 때에는 주기억기로부터 자료를 뽑아 내고 입력할 때에는 주기억기에 자료를 기억하는 역할을 한다. 다른 한가지 방법은 **직접기억기접근**(DMA)이다. 이 방식에서 I/O 모듈과 주기억기는 처리장치의 참가없이 직접 자료를 교환한다.

표 6-3 은 이 세가지 기술들의 관계를 보여 준다. 이 절에서는 프로그람식 I/O 를 고찰한다. 새치기구동식 I/O 와 DMA 는 다음의 두 절에서 각각 고찰한다.

표 6-3. I/O 기술

	새치기를 리용하지 않을 때	**새치기를 리용할 때**
처리장치를 통한 I/O 와 기억기 사이의 전송	프로그람식 I/O	새치기-구동 I/O
I/O 와 기억기사이의 직접 전송		직접기억기접근(DMA)

1. 개괄

처리장치가 프로그람을 집행하면서 I/O 에 관계되는 지령을 만나게 되면 지령을 적합한 I/O 모듈에 보내여 명령을 실행한다. 프로그람식 I/O 로 동작할 때 I/O 모듈은 필요한 동작을 수행한 다음 I/O상태등록기에 적합한 비트를 설정한다(그림 6-4). I/O모듈은 더는 처리장치에 의거하지 않는다. 특히 처리장치를 새치기하지 않는다. 그리하여 처리장치는 동작의 완료를 발견할 때까지 I/O 모듈의 상태를 주기적으로 검사해야 한다.

프로그람식 I/O 기술은 우선 처리장치가 I/O 모듈에 보내는 I/O 지령의 견지에서 고찰하고 그다음 처리장치가 실행하는 I/O 명령의 견지에서 고찰한다.

2. I/O 지령

I/O 와 관련한 명령을 실행할 때 처리장치는 특수한 I/O 모듈과 외부장치를 구별하기 위하여 주소를 내보내고 I/O 지령을 내보낸다. I/O 모듈이 처리장치에 의하여 주소화될 때 받을 수 있는 I/O 지령에는 다음의 4 가지 형태가 있다.

- **조종:** 주변장치를 동작시키는데 리용하며 해야 할 일을 지시한다. 례를 들어 자기테프장치는 테프를 감거나 레코드 앞으로 이동할데 대한 명령을 받을수 있다. 이 명령들은 적합한 형태의 주변장치에 맞는다.
- **검사:** I/O 모듈과 그 주변장치들과 관련된 각이한 상태의 조건들을 검사하는데 리용한다. 처리장치는 주목하는 주변장치에 전원이 투입되여 리용중에 있는가를 알아야 한다. 또한 가장 최근의 I/O 동작이 완료되였는가, 오유가 나타났는가 등도 알아야 한다.
- **읽기:** I/O 모듈이 주변장치로부터 자료항목을 얻어 내부완충기(그림 6-4 에서 자료등록기)에 넣도록 한다. 그 다음 처리장치는 I/O 모듈이 자료모선에 그 자료항목을 내보내도록 함으로써 자료항목을 얻을수 있다.
- **쓰기:** I/O 모듈이 자료모선으로부터 자료항목(바이트나 단어)을 꺼내고 련이어 그 자료항목을 주변장치에로 전송하게 한다.

그림 6-5 ㄱ는 주변장치로부터 기억기에로 한 블로크의 자료(즉 테프에서 한 레코드)를 읽는데 리용한 프로그람식 I/O의 실례를 보여 준다. 자료는 한번에 한 단어 (16bit)씩 읽어

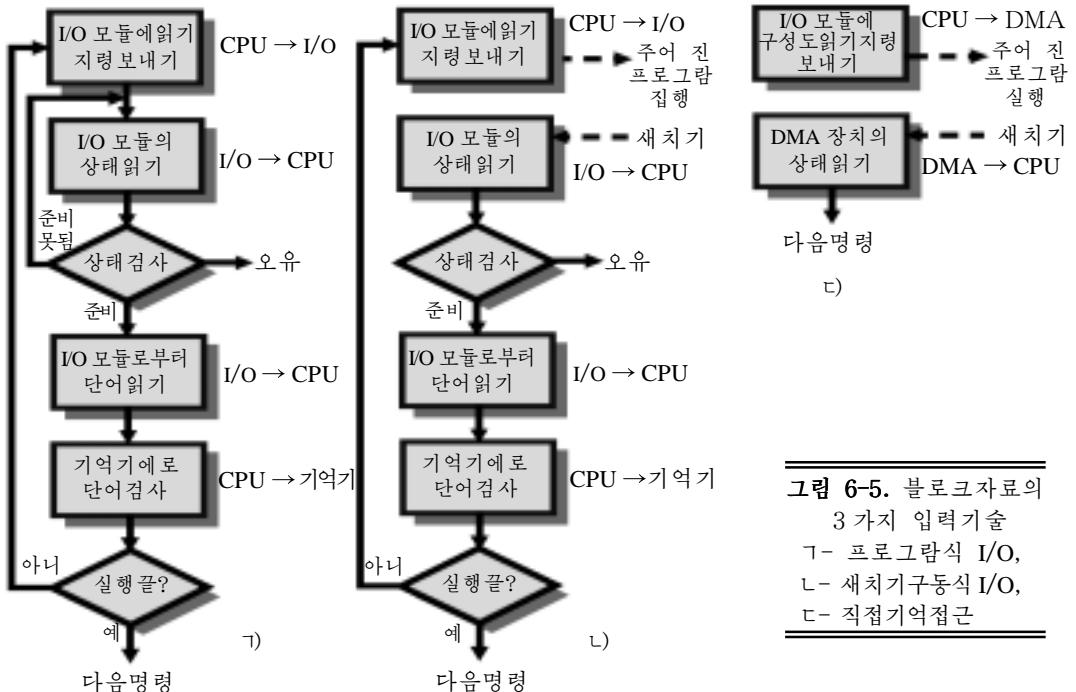



그림 6-5. 블로크자료의 3 가지 입력기술
ㄱ- 프로그람식 I/O,
ㄴ- 새치기구동식 I/O,
ㄷ- 직접기억접근

진다. 처리장치는 읽어 지는 매 단어에 대하여 그 단어가 I/O 모듈의 자료등록기에서 리용될 수 있다는것을 결정할 때까지 상태-검사주기에 놓여야 한다. 이 흐름도는 이 기술의 주요결함을 잘 보여 준다. 처리장치가 불필요한 작업을 하는것은 시간-랑비과정이다.

3. I/O 명령

프로그람식 I/O 에서 처리장치가 기억기로부터 꺼내는 I/O 관련명령들과 처리장치가 그 명령들을 실행하기 위하여 I/O 모듈에 내보내는 I/O 지령들사이에는 밀접한 대응관계가 있다. 즉 명령들은 쉽게 I/O지령으로 배치되며 이들사이에는 보통 간단한 1:1 관계가 있다. 명령형태는 외부장치들의 주소화방식에 의존한다.

일반적으로 많은 I/O 모듈들은 체계와 I/O 모듈을 통하여 련결된다. 매 장치에는 유일한 식별자나 주소가 주어 진다. 처리장치가 내보낸 I/O지령은 필요한 장치의 주소를 포함한다. 그리하여 매 I/O 모듈은 그 주소행들을 번역하여 지령이 자기것인가를 결정해야 한다.

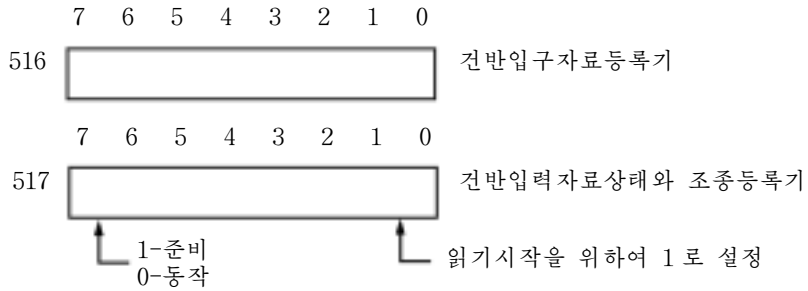
처리장치, 주기억기, I/O 가 공통모선을 공유할 때에는 두가지 방식의 주소화 즉 기억기배치와 기억기분리가 가능하다. **기억기배치식 I/O** 인 경우 기억기위치와 I/O 모듈들에는 단일한 주소공간이 있다. 처리장치는 I/O 모듈의 상태와 자료등록기를 기억기위치로 취급하고 기억기와 I/O 모듈들을 접근하는데 같은 기계명령들을 리용한다. 따라서 18 개의 주소행인 경우 임의의 조합으로 총 2^{10}=1024 개의 기억기와 I/O 주소를 얻을수 있다.

기억기배치식 I/O 에서 단일한 읽기행과 단일한 쓰기행이 모선상에서 요구된다. 다른 말로 모선에는 기억기읽기와 쓰기는 물론 입력과 출력지령행들이 설치될수 있다. 지령행은 주소가 기억기위치나 I/O 모듈을 가리키는가를 규정한다. 많은 주소령역들은 기억기위치나 I/O 모듈을 둘다 가리키는데 리용될수 있다. 10 개의 주소행이면 체계는 1024 개의 기억기위치들과 1024 개의 I/O 주소들을 다 지원할수 있다. 그것은 I/O 주소공간이 기억기의 주소공간과 분리되여 있기때문이다. 이것을 **분리식 I/O** 라고 한다.

그림 6-6 은 이 두개의 프로그람식 I/O 기술을 대비한다. 그림 6-6 ㄱ는 건반과 같은 단순한 입력장치를 위한 대면부가 기억기배치식 I/O 를 리용하는 프로그람작성자에게 어떻게 보이는가를 보여 준다. 10bit 의 주소공간을 512bit 의 기억기(위치 0～511)와 그우의 512 의 I/O 주소 (위치 512～1023)로서 가정하자. 특정한 말단으로부터의 건반입력은 두개의 주소를 리용한다. 주소 516 은 자료등록기를 가리키며 주소 517 은 상태등록기를 가리킨다. 상태등록기는 또한 처리장치지령들을 수신하기 위한 조종등록기로서도 동작한다. 프로그람은 건반으로부터 처리장치안의 축적등록기에 들어 온 1byte 의 자료를 읽는다. 처리장치는 자료바이트가 있을 때까지 이 동작을 반복한다.

분리식 I/O(그림 6-6 ㄴ)에서 I/O 포구들은 모선에서 I/O 지령행들을 능동으로 만드는 특수한 I/O 지령들에 의해서만 접근될수 있다.

대다수 형태의 처리장치들에는 기억기를 참조하기 위한 상대적으로 많은 각이한 명령모임들이 존재한다. 분리식 I/O 를 리용하면 몇개의 I/O 명령만이 있게 된다. 결국 기억기배치식 I/O 의 우점은 이 큰 명령저장고를 리용하여 더 효과적으로 프로그람을 작성할수 있다는것이다. 결함은 값 비싼 기억주소공간이 리용된다는것이다. 기억기배치식 I/O

와 분리식 I/O 는 둘다 흔히 리용된다.

주소	명령	연산수	설명
200	Load AC	“1”	
	Store AC	517	건반읽기 진행
202	Load AC	517	상태바이트 얻기
	Branch if Sign = 0	202	준비될 때까지 순환
	Load AC	516	자료바이트 얻기

ㄱ)

주소	명령	연산수	설명
200	Start I/O	5	건반읽기 진행
201	Test I/O	5	완성에 대한 검사
	Branch Not Ready	201	완성될 때까지 순환
	In	5	자료바이트 얻기

ㄴ)

그림 6-6. 기억배치식 I/O 와 분리식 I/O
ㄱ-기억배치식 I/O, ㄴ-분리식 I/O

제 4 절. 새치기구동 I/O

프로그람식 I/O 에서의 문제는 처리장치가 자료수신이나 자료송신준비에 관계하는 I/O 모듈를 오래동안 기다려야 한다는것이다. 기다리는 동안 처리장치는 I/O 모듈의 상태를 거듭 문의해야 한다. 결과 전체 체계의 성능이 대단히 떨어 진다.

다른 방법은 처리장치가 장치에 I/O 지령을 내보낸 다음 다른 유용한 작업을 계속하는것이다. I/O 모듈는 처리장치와 자료를 교환할 준비가 되면 처리장치에 새치기를 보내여 봉사를 요구한다. 그러면 처리장치는 자료전송을 실행하고 그 다음 선행한 처리를 계속한다.

우선 I/O 모듈의 관점에서 이 동작이 어떻게 진행되는가를 고찰하자. 입력할 때에 I/O 모듈는 처리장치로부터 Read 지령을 받는다. 그 다음 I/O 모듈는 련관주변장치로부터 자료를 읽는다. 일단 자료가 장치의 자료등록기에 있으면 장치는 한 조종신호선으로 처리장치에 새치기신호를 보낸다. 그 다음 장치는 처리장치가 자료를 요구할 때까지 기다린다. 요구가 접수되면 장치는 자료모선에 자료를 태우고 다른 I/O 동작을 준비한다.

처리장치의 견지에서 보면 입력동작은 다음과 같다. 처리장치는 Read 명령을 내보낸다. 그 다음 처리장치는 다른 작업을 한다(처리장치는 한번에 여러개의 각이한 프로그람을 실행할수 있다.). 매 명령주기의 끝에서 처리장치는 새치기를 검사한다(그림 3-9). I/O 모듈로부터 새치기가 발생하면 처리장치는 현재 프로그람의 상태(프로그람계수기와 처리장치등록기)를 기억시키고 그 새치기를 처리한다. 이 경우에 처리장치는 I/O 모듈로부터 자료를 읽어서 기억기에 보관한다. 그 다음 처리장치는 동작중이였던 프로그람 혹은 일부 다른 프로그람의 내용을 회복하여 실행을 계속한다.

그림 6-5 ㄴ은 한 블로크의 자료를 읽기 위한 새치기 I/O 의 리용을 보여 준다. 그림 6-5 ㄱ과 비교해 보자. 새치기 I/O 는 불필요한 대기를 없애 버리므로 프로그람식 I/O 보다 더 능률적이다. 그러나 새치기 I/O 는 여전히 처리장치시간을 많이 소비한다. 왜냐하면 기억기로부터 I/O 모듈에로 또는 I/O 모듈로부터 기억기에로 가는 모든 자료가 다 처리장치를 통과해야 하기때문이다.

1. 새치기처리

새치기구동 I/O 에서 처리장치의 역할을 더 자세히 고찰해 보자. 새치기의 발생은 처

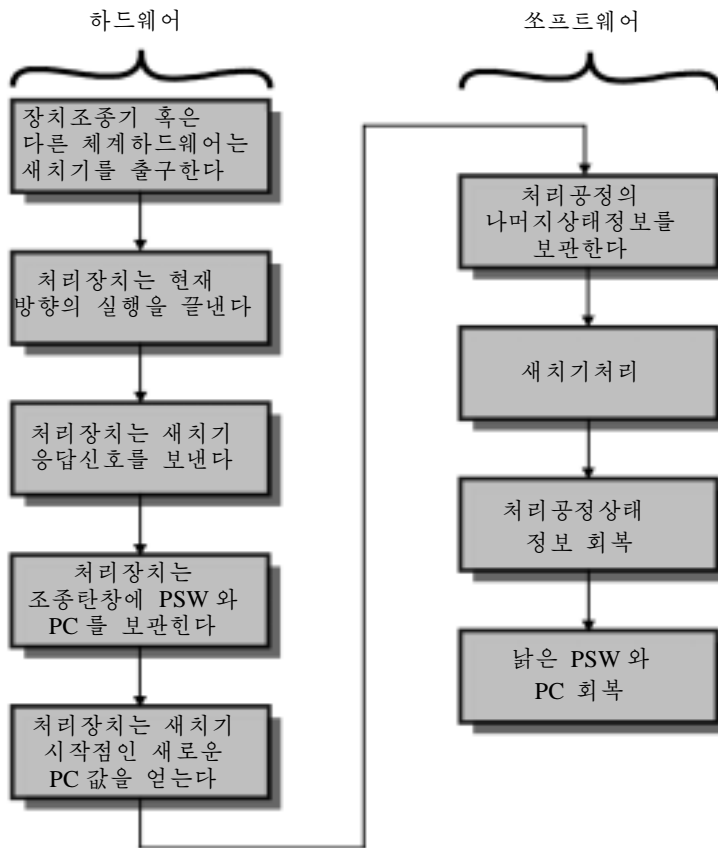



그림 6-7. 단일새치기처리

리장치하드웨어와 쏘프트웨어에서 많은 사건들을 시동시킨다. 그림 6-7 은 새치기처리의 일반적인 순서를 보여 준다. I/O 모듈이 I/O 동작을 완료할 때에는 다음과 같은 하드웨어 사건들이 발생한다.

① 장치는 처리장치에 새치기신호를 보낸다.
② 처리장치는 그림 3-9 에서 보여 준바와 같이 새치기에 응답하기전에 현재명령의 실행을 끝낸다.
③ 처리장치는 새치기가 있는가를 검사하여 새치기를 내보낸 장치에 응답신호를 보낸다. 응답을 받은 장치는 새치기신호를 없앤다.
④ 처리장치는 그 다음 새치기루틴에 대한 흐름조종준비를 요구한다. 처리장치는 시작할 때 새치기시점에서 현재 프로그람을 회복시키는데 요구되는 정보를 보관해야 한다. 요구되는 최소한의 정보는 다음과 같다.

ㄱ. 프로그람상태단어(PSW)라고 하는 등록기에 보관되는 처리장치의 상태

ㄴ. 프로그람계수기에 보관되는 실행할 다음명령의 위치

이 정보들은 체계조종탄창에 보관될수 있다.

⑤ 처리장치는 그 다음 이 새치기에 응답하는 새치기조종프로그람의 입구점위치를 프로그람계수기에 넣는다. 콤퓨터기본방식과 조작체계설계에 의거하면 모든 형태의 새치기 또는 모든 장치와 모든 형태의 새치기에 대하여 단일한 프로그람이 있을수 있다. 만일 하나이상의 새치기조종루틴이 있다면 처리장치는 어느것을 불러 내야 하는가를 결정해야 한다. 이 정보는 초기의 새치기신호에 포함되여 있을수 있다. 또한 처리장치는 새치기를 보낸 장치에 필요한 정보를 포함하는 응답을 보낼것을 요구할수 있다.

일단 프로그람계수기에 시작점위치가 설정되면 처리소자는 명령꺼내기로부터 시작되는 다음 명령주기를 수행한다. 명령꺼내기는 프로그람계수기의 내용에 의하여 결정되므로 조종권은 새치기조종프로그람에로 넘어 간다. 이 프로그람이 수행되면 다음과 같은 조작을 일으킨다.

⑥ 이 시점에서 중단된 프로그람과 관계되는 프로그람계수기와 PSW 는 체계탄창에 보관되여 있다. 그러나 집행중프로그람의 부분적인 《상태》를 보여 주는 다른 정보들도 있다. 특히 처리장치등록기의 내용들이 보관되여야 한다. 그것은 이 등록기들이 새치기조종프로그람에 의하여 리용될수 있기때문이다. 그리하여 이 모든 값들과 그외의 다른 상태정보들도 보관되여야 한다. 일반적으로 새치기조종프로그람은 탄창에 모든 등록기의 내용을 보관하는것으로부터 시작한다. 그림 6-8 ㄱ는 이에 대한 간단한 실례를 보여 준다. 이 경우에 리용자프로그람은 주소 N에 있는 명령다음에 새치기되는것으로 된다. 모든 등록기들과 다음명령의 주소 (N+1)의 내용은 탄창에 넣어 진다. 탄창지시자는 새로운 탄창끝을 가리키도록 갱신되며 프로그람계수기는 새치기봉사루틴의 시작을 가리키도록 갱신된다.

⑦ 새치기조종기는 그 다음 새치기를 처리한다. 이것은 I/O 동작과 새치기를 일으키는 다른 사건과 관계되는 상태정보에 대한 검사를 포함한다. 또한 I/O 모듈에 대한 추가적인 지령이나 응답을 보내는것도 포함할수 있다.

⑧ 새치기처리가 완료되면 보관된 등록기값들은 탄창으로부터 회복되여 등록기에 다시 넣어 진다(그림 6-8 ㄴ 참고).

⑨ 마지막으로 탄창으로부터 PSW 와 프로그람계수기값들을 회복시킨다. 결과 실행될 다음명령은 앞에서 중단된 프로그람의 명령으로 된다.

ㄱ)

ㄴ)

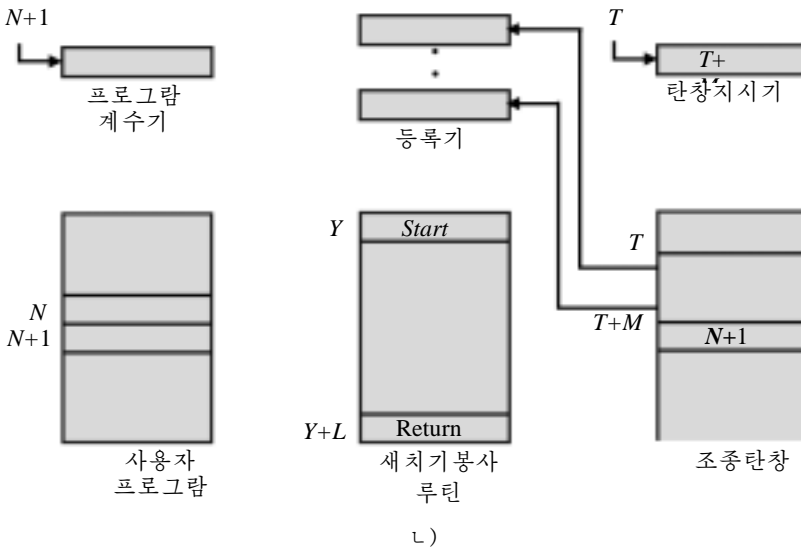
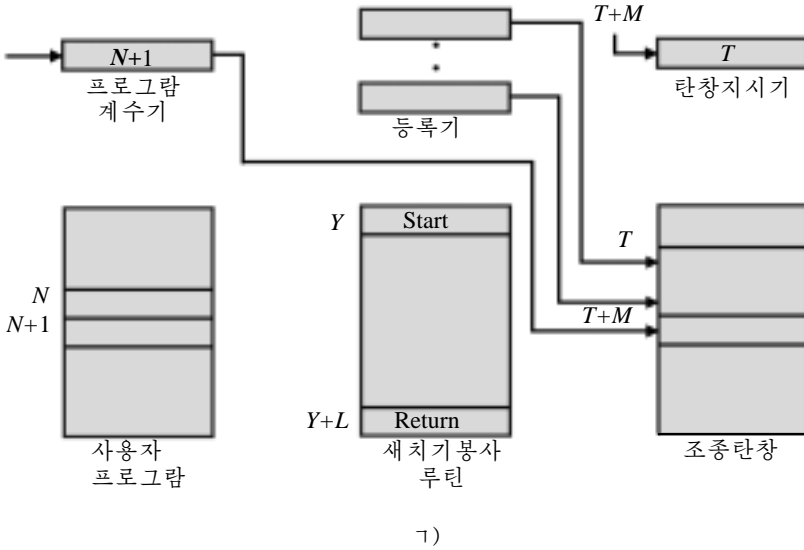



그림 6-8. 새치기처리때 기억기와 등록기의 변화
ㄱ-N 주소의 명령뒤에서 새치기발생, ㄴ-새치기로부터 되돌이

이상과 같은 고찰로부터 후에 재개(회복)를 위하여 중단된 프로그람의 모든 상태정보를 보관하는것이 중요하다는것을 강조한다. 그것은 새치기가 프로그람으로부터 접근된 루틴이 아니기때문이다. 새치기는 리용자프로그람이 실행되는 임의의 시간에 임의의 위치에서 발생할수 있다. 그의 발생은 예측할수 없다. 이 내용은 다음장에서 고찰하겠지만 그것은 두개의 프로그람이 공통적으로 일감을 가질수 없으며 서로 다른 두명의 사용자에게 속할수 있기때문이다.

2. 설계의 문제

새치기 I/O를 실현하기 위한 설계에서는 두가지 문제가 있을수 있다. 첫째로는 대체로 여러개의 I/O 모듈이 있으므로 처리장치가 새치기를 보낸 장치가 어느 장치인가를 어떻게 결정하는가 하는것이다. 둘째로 여러개의 새치기가 일어 나는 경우에 처리장치가 어느 새치기를 수행해야 하는가 하는것이다.

우선 장치를 어떻게 식별하는가를 고찰해 보자. 다음과 같은 네가지 일반적인 기술들이 공통적으로 리용된다.

- 여러개의 새치기신호선
- 쏘프트웨어문의
- 련쇄(하드웨어문의,벡토르화된)
- 모선중재(벡토르화된)

가장 간단한 방법은 처리장치와 I/O 모듈사이에 여러개의 **새치기신호선**들을 제공하는것이다. 그러나 새치기선들에 몇개 이상의 모선들이나 처리장치단자들을 전용으로 리용하는것은 비현실적이다. 뿐만아니라 가령 여러개의 선들이 리용된다고 해도 매 선은 그에 접속된 여러개의 I/O 모듈들을 가질것이다. 결국 매 선들에 이 방법을 제외한 세가지 기술들중 어느 한가지를 리용해야 한다.

다른 한가지 방법은 **쏘프트웨어문의**방법이다. 처리장치는 새치기를 검출하면 어느 장치가 새치기를 일으켰는가를 검출할수 있도록 매 I/O 모듈에 물어 보며 그의 일감은 새치기봉사루틴에로 이행한다. 이 문의는 개별적인 지령선(례로서 TESTI/O)의 형태로 될수 있다. 이 경우 처리장치는 TEST I/O를 발생하며 주소선들에 특수한 I/O 모듈의 주소를 적재한다. I/O 모듈은 새치기가 설정되면 정확히 응답한다. 즉 모든 I/O 모듈에는 주소지정을 할수 있는 상태등록기가 포함되여 있다. 처리장치는 매 I/O 모듈의 이 상태등록기를 읽어서 새치기장치를 식별한다. 일단 장치가 정확히 식별되면 처리장치는 그 장치에 적합한 특수한 장치-봉사루틴에로 이행한다.

쏘프트웨어문의방법의 결함은 시간을 소비하는것이다. 사실상 보다 효과적인 기술은 하드웨어문의을 제공하는 련쇄(daisy-chain)를 리용하는것이다. 그림 3-25 에 련쇄구조의 실례를 보여 주었다. 이 방법에서는 새치기에 대하여 모든 I/O 모듈이 공통새치기요구선을 공유한다. 새치기응답선은 장치를 통하여 련속적으로 련결되여 있다. 처리장치는 새치기를 수감하면 새치기응답을 보낸다. 이 응답신호는 알맞는 장치에 도달할 때까지 I/O 모듈들을 통과한다. 새치기요구장치는 대체로 자료선들에 한 단어를 태우는것으로 이에

응답한다. 이 단어는 벡토르로서 I/O 모듈의 주소이거나 혹은 다른 유일한 식별자일수도 있다. 이 두 경우에 처리장치는 벡토르를 적절한 장치-봉사루틴에로의 지적자로 리용한다. 이렇게 하면 일반적인 새치기-봉사루틴을 실행하지 않아도 된다. 이 기술을 **벡토르화된 새치기**라고 한다.

벡토르새치기를 리용하는 또 다른 하나의 기술은 **모선중재**이다. 모선중재에서 I/O 모듈는 우선 새치기요구선에 새치기요구신호를 태우기전에 모선조종권을 획득해야 한다. 결국 한번에 하나의 장치만이 그 선에 새치기요구신호를 태울수 있다. 처리장치가 새치기를 검출하면 새치기응답선으로 이에 응답한다. 그 다음 새치기를 요구한 장치는 그의 벡토르를 자료선에 태운다.

이상에서 언급한 기술들은 새치기를 요구한 I/O 모듈을 식별하는데 리용한다. 또한 하나이상의 장치가 새치기봉사를 요구할 때 우선권을 할당하는 방법을 제공한다. 여러개의 선인 경우 처리장치는 가장 높은 우선권을 가진 새치기선을 포착한다. 쏘프트웨어문의에서는 장치들에 물어 보는 순서에 따라 우선권이 결정된다. 이와 류사하게 련쇄방법에서도 장치들의 순서 즉 우선권을 결정한다. 끝으로 모선중재는 제3장 4절에서 론의한 바와 같이 우선권방식을 리용할수 있다.

이제부터 새치기구조에 대한 두가지 실례를 보기로 하자.

3. Intel 82C59A 새치기조종기

Intel 80386 은 단일한 새치기요구(INTR)와 단일한 새치기응답(INTA)선을 가지고 있다. 80386 이 각이한 종류의 장치들과 우선권구조들을 조종하게 하자면 보통 82C59A 와 같은 외부새치기중재기를 리용해야 한다. 외부장치들은 82C59A 에 련결되며 그 다음에 이를 통하여 80386 과 련결된다.

그림 6-9 는 80386 에 여러개의 I/O 모듈를 련결하는 82C59A 의 리용을 보여 준다. 하나의 82C59A 는 8 개까지의 장치를 조종할수 있다. 8 개이상의 장치를 조종하는 경우에는 82C59A 새치기조종기를 종속련결하여 64 개까지의 장치를 조종할수 있다.

82C59A 의 유일한 역할은 새치기관리이다. 82C59A 는 련결된 장치로부터 새치기요구를 접수하여 어느 새치기가 가장 높은 우선권을 가지는가를 결정한 다음 INTR 선을 통하여 처리장치에 신호한다. 처리장치는 INTA 선을 통하여 응답한다. 이것은 82C59A 로 하여금 자료모선에 해당한 벡토르정보를 내보낸다. 처리장치는 그 다음 새치기처리를 진행할수 있으며 자료를 읽거나 쓰기 위하여 I/O 모듈과 직접 통신할수 있다.

82C59A 는 프로그람화할수 있다. 80386 은 82C59A 에 조종단어를 설정하여 리용되는 우선권을 결정한다. 다음과 같은 새치기방식들이 있다.

- **고정순위방식:** 새치기요구들이 0(IR0)부터 7(IR7)까지의 우선권으로 등급화된다.
- **회전순위방식:** 일부 응용들에서 많은 새치기장치들은 우선권이 같다. 이 방식에서 한 장치는 봉사된후에 그 그룹에서 가장 낮은 우선권을 가진다.
- **특수한 금지:** 이것은 처리장치가 어떤 장치로부터의 새치기를 금지하도록 한다.

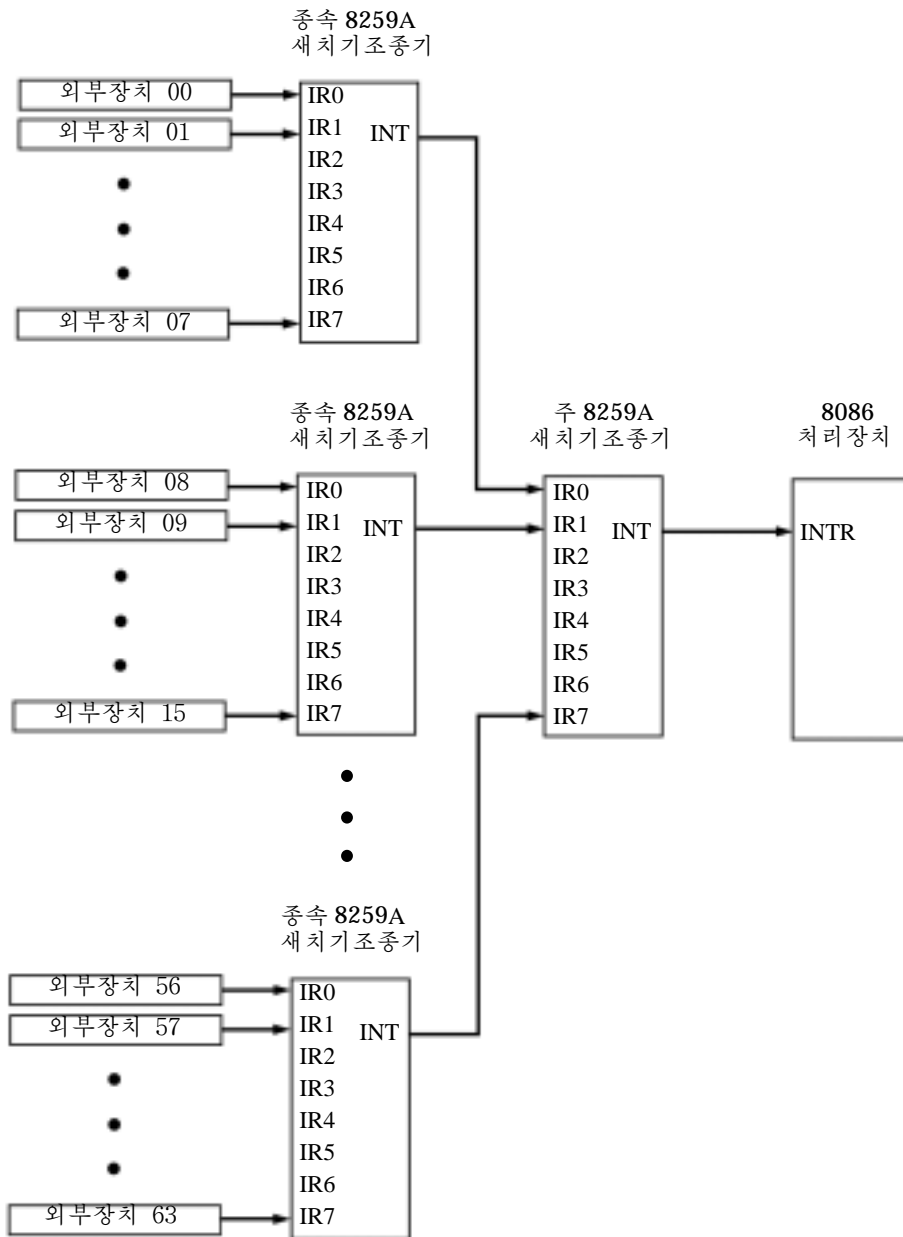



그림 6-9. 82C59A 새치기조종기의 리용

4. Intel 82C55A의 프로그람가능 주변장치대면부

프로그람식 I/O와 새치기-구동I/O에서 리용된 I/O모듈의 실례로서 여기서는 Intel 82C55A 의 프로그람가능 주변장치대면부를 고찰한다. 82C55A 는 단일소편이며 Intel 80386 처리장치용으로 설계된 일반목적 I/O 모듈이다. 그림 6-10 은 일반적인 구성도와 40

개의 단자로 된 소자의 다리배치도를 보여 주었다.

구성도에서 오른쪽은 82C55A 의 외부대면부이다. 80386 은 조종등록기를 리용하여 24 개의 I/O 선들을 프로그람화할수 있다. 80386 은 여러가지 동작방식들과 입출구형태들을 규정하기 위하여 조종등록기에 값을 설정한다. 24 개의 선들은 세개의 8bit 그룹(A, B, C)들로 나누어 진다. 매 그룹은 8bit 의 I/O 포구로 리용할수 있다. 더우기 그룹 C 는 A 와 B 의 I/O 포구들과 결합하여 리용할수 있는 4bit 의 그룹들(C_A 와 C_B)로 나누어 진다. 24 개의 선들은 이와 같은 방법으로 구성되여 조종 및 상태신호들을 전송한다.

구성도에서 왼쪽은 80386 모선과의 내부대면부이다. 여기에는 I/O 포구와 자료를 주고 받으며 조종정보를 조종등록기에로 전송하는데 리용하는 8bit 쌍방향자료모선(D0~D7)이 있다. 두개의 주소선들은 3 개의 I/O 포구와 조종등록기중 어느 하나를 지적한다. 전송은 READ 혹은 WRITE 선들과 함께 CHIP SELECT 선이 능동상태로 되였을 때 진행된다. RESET 선은 장치를 초기화하는데 리용된다.

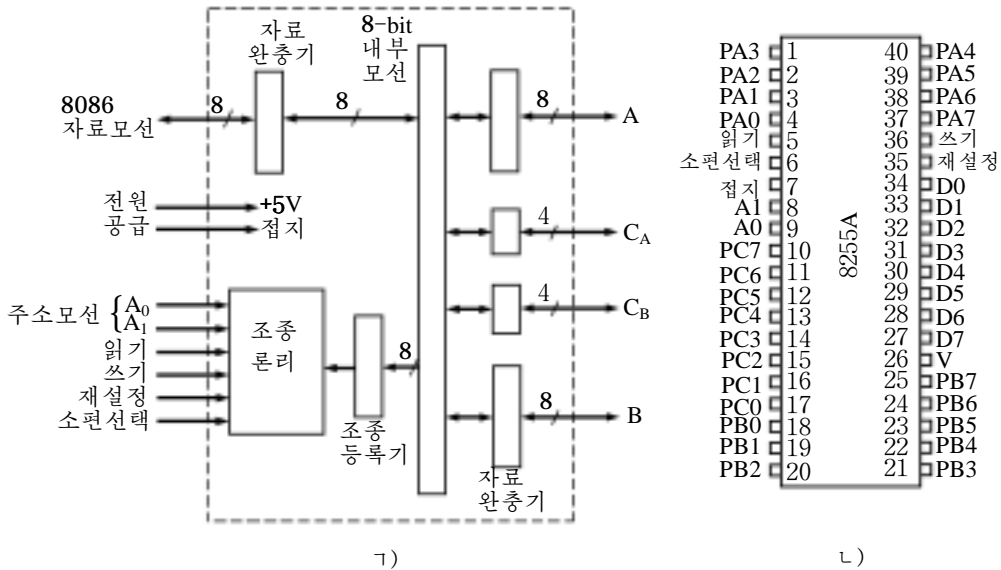



그림 6-10. Intel 82C55A 프로그람 가능한 주변결합소자
ㄱ- 블로크도, ㄴ- 단자배치도

처리장치는 동작방식을 조종하며 신호들을 정의하기 위하여 조종단어를 임의로 조종등록기에 넣는다. 3 개 그룹의 8 개의 외부선들은 3 개의 8bit I/O 포구로 작용한다. 매 포구는 입구 혹은 출구로 지정될수 있다. 한편 그룹 A 와 B 는 I/O 포구로서 리용하며 그룹 C 의 선들은 그룹 A 와 B 의 조종선들로 리용한다. 조종신호선들은 두가지 기본목적 즉 “맞잡이조종방식”과 새치기요구에 쓰인다. 맞잡이조종방식은 단순한 시간일치기구이다. DY 선으로서 송신자에 의하여 리용되며 자료가 I/O 자료선에 있다는것을 지적한다. 또 다른 선은 ACKNOWLEDGE 선으로서 수신자에 의하여 리용되며 자료를 읽었으므로

자료선들에 아무 자료도 없다는것을 지적한다. 또 다른 선은 INTERRUPT REQUEST 선으로서 이 선은 체계모선에 구속련결되여 있다.

82C55A 는 조종등록기를 통하여 프로그람화할수 있으므로 여러가지 간단한 주변장치들을 조종하는데 리용할수 있다. 그림 6-11 은 건반/현시장치 말단을 조종하기 위해 리용된 82C55A 를 보여 준다. 건반은 8bit 입력을 제공한다. 이 중 두개의 비트 즉 SHIFT 와 CONTROL 은 처리장치에서 실행하고 있는 건반-조종프로그람에서 특별한 의미를 가진다. 이 설명은 단순히 8bit 자료를 접수하여 그것을 체계자료모선에 태우는 82C55A 에서는 명백하다. 두개의 맞잡이조종선들은 건반리용을 위해 제공되는 선들이다.

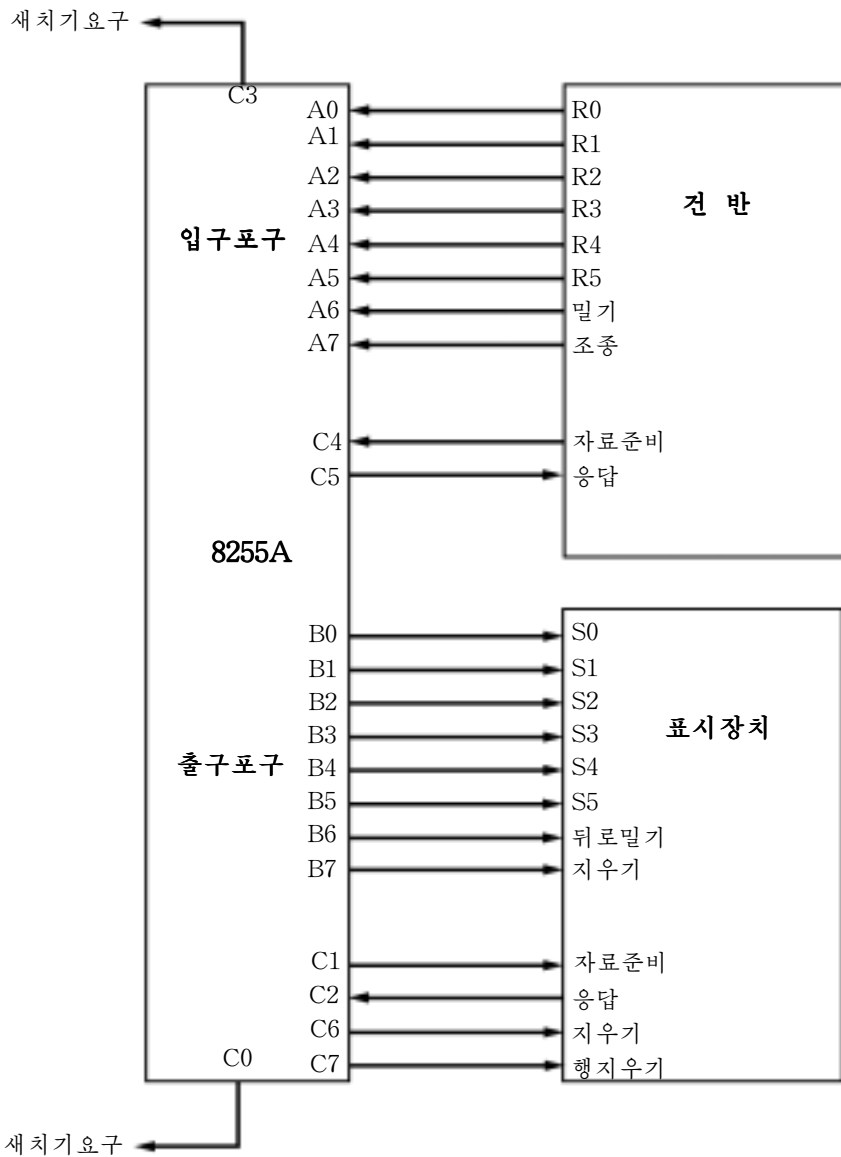



그림 6-11. 8255C 에 의한 건반/표시장치대면부

현시장치는 또한 8bit 자료포구와 련결된다. 이 비트들중 두개의 비트 역시 82C55A 에서 명백한,특수한 의미를 가진다. 두개의 맞잡이조종선들외에 두개의 선들이 추가적인 조종기능을 제공한다.

제 5 절. 직접기억기접근

1. 프로그람식 I/O 와 새치기구동 I/O 의 결함

새치기구동 I/O 는 단순한 프로그람식 I/O 보다 더 효과적이지만 여전히 기억기의 I/O 모듈사이에서 자료를 전송하자면 처리장치의 능동조정을 요구한다. 또한 자료전송은 처리장치를 통하여서만 진행되여야 한다. 따라서 이 두 방식은 그 자체의 고유한 다음과 같은 두가지 결함을 가지고 있다.

1. I/O 전송속도는 처리장치가 장치를 검사하고 그 장치에 봉사할수 있는 속도에 의하여 제한된다.
2. 처리장치는 매 I/O 전송에서 많은 지령들이 실행되여야 하므로 I/O 전송관리에 구속된다(례 그림 6-5).

이 두가지 결함들사이에는 일련의 용납되지 않는 점들이 있다. 이제 블로크자료에 대한 전송을 고찰해 보자. 단순한 프로그람식 I/O 를 리용하는 경우 처리장치는 I/O 과제만을 수행하므로 그외의 다른 일을 또 할 때보다 오히려 높은 속도로 자료를 전송할수 있다. 새치기 I/O 는 이 I/O 전송속도에서 처리장치를 어느 정도 해방한다. 그럼에도 불구하고 이 두가지 방법들은 모두 처리장치동작과 I/O 전송속도에 나쁜 영향을 준다.

큰 자료를 전송할 때에는 더 효과적인 기술 즉 직접기억기접근(DMA)을 리용한다.

2. DMA 기능

DMA 는 체계모선에 추가적장치를 포함시킬것을 요구한다. DMA 장치(그림 6-12)은 처리장치로 되려는 능력 즉 처리장치로부터 체계에 대한 조종을 실현할 능력이 있다. 이것은 체계모선을 통하여 기억기와 자료를 전송할것을 요구한다. 이를 위하여 DMA 장치는 처리장치가 요구하지 않을 때에만 모선을 리용하여야 한다. 또는 처리장치가 일시적으로 동작을 지연시키도록 하여야 한다. 이 기술이 더 일반적으로 리용되며 DMA 장치가 효과적으로 모선주기를 훔치므로 **주기훔침방법**(cycle stealing)이라고 한다.

처리장치가 한 블로크의 자료를 읽거나 쓸것을 요구한다면 다음과 같은 정보를 보내여 DMA 장치에 지령을 준다.

- 처리장치와 DMA 장치사이의 읽기나 쓰기조종선을 리용하여 읽기나 쓰기를 요구한다.
- 자료선들을 리용하여 전송되는 I/O 모듈의 주소

• 자료선우에 전송되고 DMA 장치에 의하여 그의 주소등록기에 기억되는 읽기나 쓰기를 위한 기억기의 시작주소.
• 자료선들을 통하여 다시 전송되며 자료계수등록기에 기억되는 읽기나 쓰기를 위한 단어수.

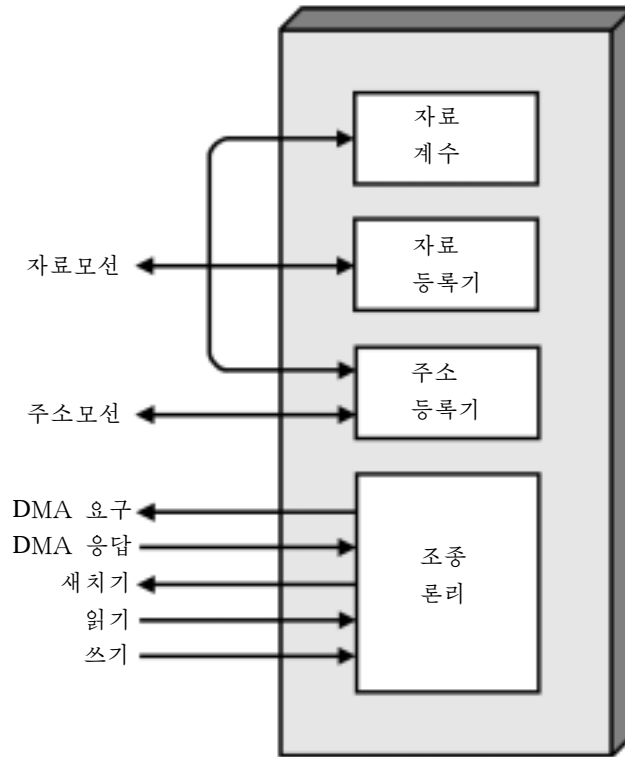



그림 6-12. 일반적인 DMA 구성도

처리장치는 그 다음 다른 동작을 계속한다. 처리장치는 이 I/O동작을 DMA장치에 넘겨 준다. DMA 장치는 처리장치의 참가 없이 전체 자료블로크를 한번에 한 단어씩 기억기로부터 기억기에로 직접 전송한다. 전송이 완료되였을 때 DMA 장치는 처리장치에 새치기신호를 보낸다. 그리하여 처리장치는 전송의 시작과 끝에서만 참가한다(그림 6-5 ㄷ). 그림 6-13 은 명령주기의 어느 위치에서 처리장치가 중지될수 있는가를 보여 준다. 매 경우에 처리장치는 모선을 리용하기 직전에 중지된다. DMA 장치는 그 다음 한 단어를 전송하고 처리장치에 조종권을 넘겨 준다. 이것은 새치기가 아니다. 즉 처리장치는 현상태를 보관하지 않고 다른 일을 한다. 오히려 처리장치는 한 모선주기동안 일시정지(pause)한다. 결국 처리장치가 더 느리게 동작하게 된다. 그럼에도 불구하고 다중단어 I/O 전송에서 DMA 는 새치기구동이나 프로그람식 I/O 보다 훨씬 더 효과적이다.

DMA 기구는 여러가지 방법으로 구성할수 있다. 그 일부를 그림 6-14에 보여 준다. 첫 실례에서 모든 장치들은 같은 체계모선을 공유한다. DMA 장치는 대리처리장치로서 동작하면서 DMA 장치를 통하여 기억기와 I/O 모듈사이에서 자료를 교환하기 위하여 프로그람식 I/O를 리용한다. 이 구성은 원가가 눅지만 효률이 낮다. 처리장치-조종프로그람식 I/O에서 한 단어의 전송은 두개의 모선주기를 소비한다.

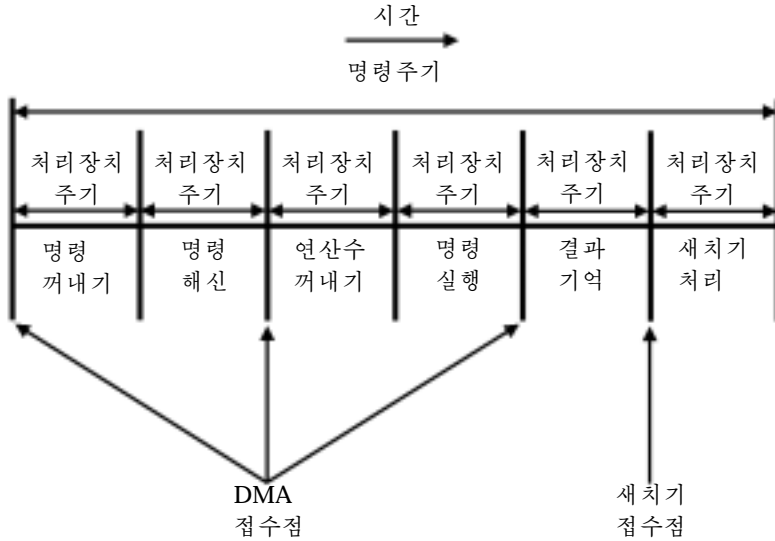



그림 6-13. 명령주기안에서 DMA와 새치기 접수점

요구되는 모선주기의 수는 DMA와 I/O 함수들을 통합함으로써 실질적으로 잘리워 질 수 있다. 그림 6-14 ㄴ가 보여 주는바와 같이 이것은 DMA 장치와 체계모선을 포함하지 않는 하나이상의 I/O 모듈사이에 통로가 있다는것을 의미한다. DMA 론리는 실제상 I/O 모듈의 부분일수 있다. 또는 하나이상의 I/O 모듈들을 조종하는 분리된 장치일수 있다. 이 개념은 I/O 모선을 리용하여 I/O 모듈을 DMA 장치에 련결시킴으로써 한 단계 더 높은것으로 된다. 이것은 DMA 장치에서 I/O 대면부들의 수를 하나로 줄이며 쉽게 확장할수 있는 구성을 제공한다. 이 모든 경우(그림 6-14 ㄴ와 6-14 ㄷ)들에서 DMA 장치가 처리장치와 기억기와 함께 공유하는 체계모선은 기억기와 자료를 교환할 때에만 DMA 장치에 의하여 리용된다. DMA와 I/O 모듈사이에서의 자료교환은 체계모선에서 발생한다.

제 6 절. I/O 통로와 처리장치

1. I/O 기능의 진화

콤퓨터체계가 진화하면서 개별적구성요소들의 복잡성과 성능이 높아 지게 되였다. 이것은 I/O 기능에서 더 명백히 나타난다. 앞에서 이미 진화과정의 한 부분을 보았다.

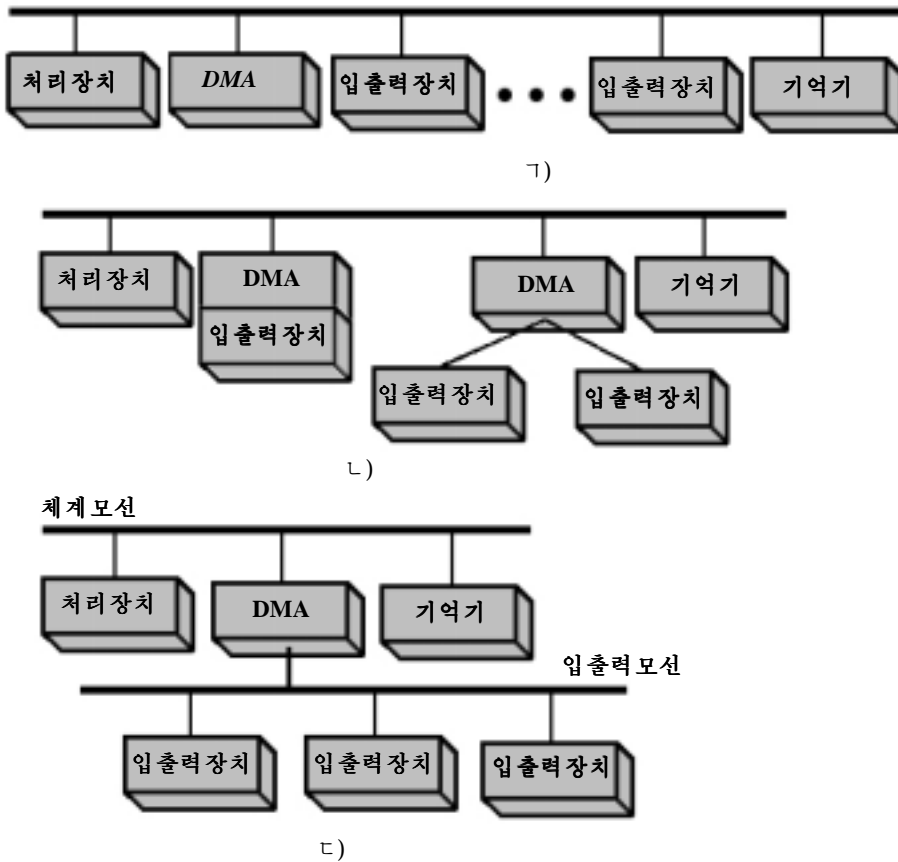



그림 6-14. 여러가지 DMA 구성
ㄱ-단일모선, 분리된 DMA, ㄴ-단일모선, 묶어 진 DMA-입출력, ㄷ-입출력모선

진화단계는 다음과 같이 요약할수 있다.

1. CPU 는 주변장치를 직접 조종한다. 이것은 간단한 극소형처리장치-조종장치들에서 볼수 있다.
2. 조종기나 I/O 모듈이 추가된다. CPU 는 새치기가 없는 프로그람식 I/O 를 리용한다. 이 단계에서 CPU 는 구체적인 외부장치대면부와 분리된다.
3. 단계 2 에서와 같은 구성이 리용되지만 여기서는 새치기가 리용된다. CPU 는 I/O 동작수행대기에 시간을 소비하지 않고 효률을 높인다.
4. I/O 모듈은 DMA 를 통하여 기억기직접접근을 수행한다. 이것은 전송의 시작과 끝을 제외하고는 CPU 의 참가없이 자료블로크를 기억기로, 기억기에서 이동시킬수 있다.
5. I/O 모듈은 그 자체가 I/O 를 위해 만들어 진 특수한 명령묶음을 가진 처리장치의 역할을 하도록 하여야 한다. CPU 는 기억기에 있는 I/O 프로그람을 수행하도록 I/O 처리장치에 지령한다. 이것은 CPU 가 I/O 동작의 순서를 규정하게 하며

전체 순서가 수행될 때에만 새치기되도록 한다.

6. I/O 모듈은 그 자체의 국부기억기를 가진다. 사실상 그 자체가 콤퓨터라고 할수 있다. 이 구성방식에서 많은 I/O 모듈들이 최소한 CPU 의 참가밑에 조종될수 있다. 그러한 기본방식에 대한 일반적리용은 호상작용하는 말단들과의 조종통신이다. I/O 처리장치는 말단들을 조종하는데 포함된 많은 파일들을 관리한다.

콤퓨터체계가 이 경로를 따라 발전하기때문에 점점 더 많은 **I/O 동작**들이 CPU 참가없이 수행된다. CPU 는 I/O 와 관련과제에서 크게 면제되므로 성능을 개선하고 있다. 마지막단계들(5 와 6)에서는 프로그람을 실행하는 능력을 가진 I/O 모듈에 대한 개념이 도입됨으로 하여 주요변화가 일어 났다. 단계 5 에서 I/O 모듈은 흔히 **I/O 통로**로 지적된다. 단계 6 에서 **I/O 처리장치**라는 용어가 흔히 리용된다. 그러나 두가지 용어들은 두가지 상황에 다 적응된것을 전제로 한다. 이제부터는 I/O 통로라는 용어를 리용한다.

2. I/O 통로의 특징

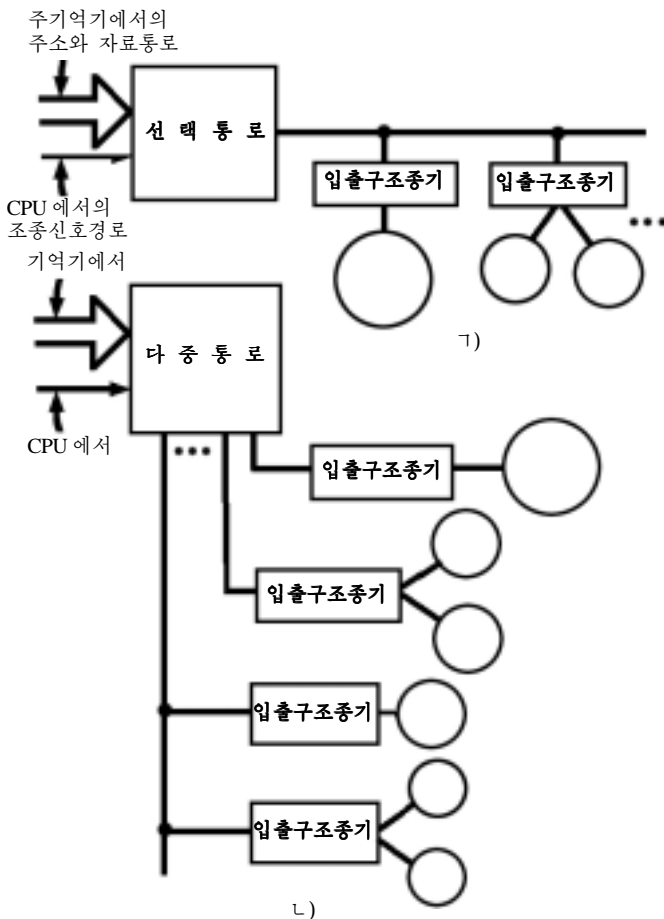



그림 6-15. I/O 통로기본방식
ㄱ-선택통로에 의한 구성, ㄴ-다중통로에 의한 구성

I/O 통로는 DMA 개념의 확장이다. I/O 통로는 I/O 동작을 완전히 조종하는 I/O 명령들을 실행할 능력이 있다. 그러한 장치들을 가진 콤퓨터체계에서 CPU 는 I/O 명령들을 실행하지 않는다. 그러한 명령들은 주기억기에 보관되며 I/O 통로 그 자체에 있는 전문목적처리장치에 의하여 실행된다. 그리하여 CPU 는 I/O 통로에 주기억기에 있는 프로그람을 실행할것을 명령함으로써 I/O 전송을 보장한다. 프로그람은 장치나 장치들, 대용량기억기령역, 우선권, 어떤 오유조건에 해당한 작용들을 밝힌것이다. I/O 통로는 이 명령들에 의해 자료전송을 조종한다.

그림 6-15 에서 보여 주는 두가지 형태의 I/O 통로들이 일반적으로 리용된다. **통로선택기**는 여러가지의 고속장치들을 조종하며 한번에 그 장치들중 하나와 자료전송한다. 그러므로 I/O 통로

는 한 장치를 선택하고 자료전송에 영향을 준다. 매 장치,또는 적은 묶음의 장치들은 **조종기**나 우리가 론의한 I/O 모듈과 거의 같은 I/O 모듈에 의하여 조종된다. 그러므로 I/O 통로는 이 I/O 조종기들을 조종하는 CPU 를 대신하여 리용된다. **다중화통로**는 단번에 여러개 장치들의 I/O 를 조종할수 있다. 저속장치들에서 한 **바이트다중화장치**는 가능한껏 빨리 문자들을 접수하거나 전송한다. 례를 들어 속도가 각이하며 개별적흐름이 각각 $A_1A_2A_3A_4\cdots$, $B_1B_2B_3B_4\cdots$, $C_1C_2C_3C_4\cdots$인 세개의 장치들로부터 오는 결과적인 문자흐름은 $A_1B_1C_1A_2C_2A_3B_2C_3A_4\cdots$이다. 고속장치들에서는 **블로크다중화장치**가 여러 장치들로부터 오는 자료블로크를 교차읽기쓰기한다.

제 7 절. 외부대면부: SCSI 와 FireWire

1. 대면부의 형태

I/O 모듈로부터 주변장치에로의 대면부는 주변장치의 특징과 동작에 맞게 제작되여야한다. 대면부의 한가지 주요특징은 직렬인가 병렬인가 하는것이다(그림 6-16). **병렬대면부**에는 I/O 모듈과 주변장치들을 련결하는 여러개의 선들이 있으며 한 단어의 모든 비트들이 동시에 전송되는것처럼 자료모선을 통하여 여러개의 비트들이 동시에 전송된다. **직렬대면부**에서는 자료전송에 한개의 선만이 리용되며 한번에 한바이트씩 전송된다. 병렬대면부는 일반적으로 테프, 디스크와 같은 고속대면부들에 리용된다. 직렬대면부는 인쇄기와 말단에 더 적합하다.

이 두 경우에 I/O 모듈은 주변장치와 대화를 진행해야 한다. 일반적으로 쓰기동작의 대화는 다음과 같다.

- I/O 모듈은 자료를 보낼것을 요구하는 조종신호를 보낸다.
- 주변장치는 그 요구에 응답한다.
- I/O 모듈은 자료를 전송한다(주변장치에 따르는 한 단어나 블로크).
- 주변장치는 자료의 수신에 응답한다.

읽기동작도 류사하게 진행된다.

I/O 모듈의 동작에서 기본역할을 하는것은 주변장치와 체계의 나머지부분들사이에서 주고 받는 자료를 기억할수 있는 내부완충기이다. 이 완충기는 I/O 모듈가 체계모선과 외부선들사이에서 속도상 차이를 없애 버린다.

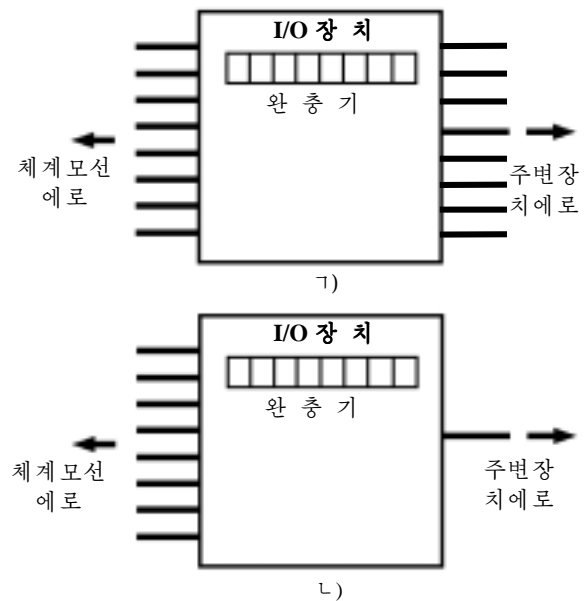



그림 6-16. 병렬과 직렬 I/O
ㄱ-병렬 I/O, ㄴ-직렬 I/O

2. 1대 1및 1대 N구조

콤퓨터체계에 있는 I/O 모듈과 외부장치들사이의 련결은 1대 1 및 1대 N일수 있다. 1대 1 대면부는 I/O 모듈과 외부장치사이에 있는 선을 리용한다. 소규모체계들(개인용콤퓨터, 워크스테이션)에서 대표적인 1대 1 련결은 건반, 인쇄기, 외부모뎀들을 포함한다. 그러한 대면부의 전형적인 실례는 EIA-232 이다(이에 대해서는 [STAL97]을 참고).

중요한것은 1 대 N 외부대면부이며 이것은 외부대용량기억장치(디스크와 테프구동기)와 다매체장치들(CD-ROM, 비데오, 음성장치)에 리용된다. 이 1대 N 대면부들은 효과적인 외부모선들이다. 또한 제 3장에서 론의된 모선들과 같은 형의 론리를 가진다. 이 절에서는 두가지 주요실례들인 SCSI와 FireWire를 보게 된다.

3. 소형콤퓨터체계대면부

SCSI는 외부주변장치들의 대표적인 실례이다. 이것은 1984년 마킨토쉬에 처음으로 도입되였다. SCSI는 현재 많은 워크스테이션에서는 물론 마킨토쉬와 Windows/Intel 체계들에서 널리 리용되고 있다. SCSI는 CD-ROM 구동기, 음성장치, 외부대용량기억장치들에 대한 표준대면부이다. SCSI는 8, 16, 32개의 자료선들을 가진 병렬대면부를 리용한다.

사실상 장치들이 서로 련속적으로 련결되여 있어도 SCSI 구성은 보통 모선으로서 지적된다. 매 SCSI장치에는 입구, 출구의 두개의 접속기가 있다. 모든 장치들은 서로 사슬처럼 련결된다. 사슬의 한 끝은 주콤퓨터에 련결되여 있다. 모든 장치들은 독립적으로 동작하며 주콤퓨터와는 물론 서로 자료를 교환할수 있다. 례를 들어 하드디스크는 주콤퓨터의 참가없이 자체로 테프구동기에 재복사될수 있다. 앞으로 서술되겠지만 자료는 통보문파케트로 전송된다.

SCSI판본들

초기의 SCSI는 SCSI-1이라고 하며 1980년대초에 개발되였다. SCSI-1은 8개의 자료선을 리용하며 5MHz의 박자속도나 5Mbyte/s의 자료속도로 동작한다. SCSI-1 은 7개까지의 장치를 종속련결시켜 주체계에 걸어 준다.

1991년에 교정판 SCSI-2가 소개되였다. 가장 주목할만한 변화는 16개나 32개까지로 자료선들을 선택적으로 확장한것이며 박자속도를 10MHz까지 증가시킨것이다. 결과 최대자료속도가 20혹은 40Mbyte/s로 되였다. 현재 SCSI-3이 개발중에 있으며 이것은 더 큰 속도를 보장할것이다.

신호와 시간단계

SCSI 모선에서의 모든 교환은 발신자(발신자)와 수신자 혹은 목적지(수신자)사이에 진행된다. 일반적으로 주체계는 발신자이며 주변조종소자는 수신자이다. 그러나 일부 장치들은 둘 중 어느 한가지 역할을 할수 있다. 임의의 경우에 모선우에서의 모든 작용은 련속적인 시간단계들에서 일어 난다. 그 시간단계들은 다음과 같다.

- **모선 비여있기:** 어떤 장치도 모선을 리용하지 않으며 모선이 리용될수 있다는것을 지

적한다.

- **중재:** I/O 처리를 초기화하거나 재개하도록 한 장치가 모선조종권을 획득하도록 한다.
- **선택:** 발신자가 읽기나 쓰기지령과 같은 기능을 수행하도록 수신자를 선택할수 있게 한다.
- **재선택:** 수신자가 발신자에 의하여 이미 시동되였으나 수신자에 의하여 중지된 조작을 회복하기 위하여 발신자와 다시 련결되게 한다.
- **지령:** 수신자가 발신자로부터 지령정보를 요구하도록 한다.
- **자료:** 수신자가 자료전송을 발신자로부터 수신자에로 즉 자료받기 또는 수신자로부터 발신자에로 즉 자료보내기를 요구하게 한다.
- **상태**: 수신자가 수신자로부터 발신자에로 보낸 상태정보를 요구하게 한다.
- **통보**: 수신자로부터 발신자에게로 또는 발신자로부터 수신자에게로 하나이상의 통보문을 보낼것을 수신자가 요구할수 있게 한다.

그림 6-17은 SCSI 모선시간단계들이 일어 나는 순서를 보여 준다. 재설정 혹은 전원을 넣은 다음 모선은 모선이 비여 있는 단계에 들어 간다. 다음 중재단계가 뒤따른다. 보통 이 단계는 어떤 장치에 의하여 조종권을 얻는다. 중재단계가 끝나면 모선은 모선빔단계에로 되돌아 간다. 중재가 성공하면 모선은 이 교환을 위하여 발신자와 수신자장치들을 할당하는 선택이나 재선택단계에 들어 간다. 두 장치가 결정된후에는 이 두 장치들이 동반되는 하나이상의 정보전송단계들(지령, 자료, 상태, 통보)이 있게 된다.

마지막정보전송단계는 일반적으로 통보받기단계이다. 이 단계에서는 차단이나 지령완료통보가 발신자에로 전송되며 모선이 비여 있는 단계가 뒤따르게 된다.

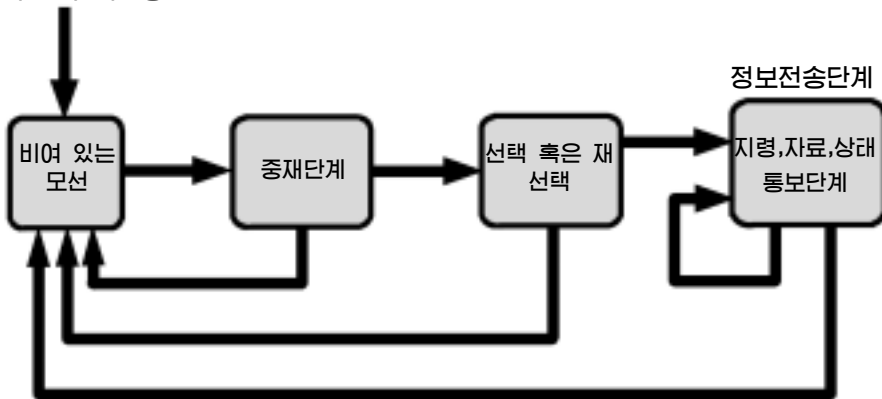



그림 6-17. SCSI 모선단계

SCSI의 한가지 중요한 특징은 재선택능력이다. 완료에 일정한 시간이 걸리는 지령이 출력되면 수신자는 모선을 해방하고 그 다음 발신자와 다시 련결될수 있다. 례를 들어 주콤퓨터는 디스크구동기에 디스크를 형식화할데 대한 지령을 줄수 있으며 구동기는

모선을 구속하지 않고 동작을 수행할수 있다.

SCSI-1 은 18 개의 신호선들로 이루어 진 케블을 정의한다. 여기서 9 개는 조종선이고 9 개는 자료선(8 개는 자료선, 1 개는 기우성선)이다. 조종선들은 다음과 같다.

- **BSY:** 발신자나 수신자에 의하여 설정되여 모선이 작업중이라는것을 지적한다.
- **SEL:** 발신자에 의하여 리용되여 지령을 수행할 수신자를 선택하거나 또는 수신자에 의하여 리용되여 그것이 차단되는 발신자를 다시 선택한다. I/O 신호는 선택과 재선택사이를 구별한다.
- **C/D:** 수신자에 의하여 리용되는 자료모선우의 자료가 조종(지령, 상태 또는 통보문) 정보인가 자료정보인가를 지적한다.
- **I/O:** 수신자에 의하여 리용되며 자료모선우에서의 자료이동방향을 조종한다.
- **MSG:** 수신자에 의하여 리용되며 전송되고 있는 정보가 통보문이라는것을 발신자에게 알려 준다.
- **REQ:** 수신자에 의하여 리용되며 자료정보전송을 요구한다. REQ 신호에 응답하여 발신자는 자료받기단계기간에 모선으로부터 자료를 접수하거나 자료보내기단계기간에 모선에 정보를 보낸다.

표 6-4. SCSI 모선신호

모선시간 주기	A 케블신호					B 케블신호		
	BSY	SEL	C/D I/O MSG REQ	ACK ATN	DB(7-0) DB(P)	REQP	ACKB	DB(31-8) DB(P1) DB(P2) DB(P3)
모선빔	None	None	None	None	None	None	None	None
중재	All	Win	None	None	S ID	None	None	None
선택	I&T	Init	None	Init	Init	None	None	None
재선택	I&T	None	Targ	Init	Targ	None	None	None
지령	Targ	None	Targ	Init	Init	None	None	None
자료받기	Targ	None	Targ	Init	Targ	Targ	Init	Targ
자료보내기	Targ	None	Targ	Init	Init	Targ	Init	Init
상태	Targ	None	Targ	Init	Targ	None	None	None
통보받기	Targ	None	Targ	Init	Targ	None	None	None
통보보내기	Targ	None	Targ	Init	Init	None	None	None

ALL: 신호는 능동적으로 중재되는 모든 SCSI 장치들에 의하여 구동된다.
SID: 유일자료비트(SCSI ID)는 능동적으로 중재되는 모든 장치들에 의하여 구동된다.
I &T: 신호는 선택과 재선택단계에서 구별되는것처럼 발신자, 수신자 혹은 이 둘에 의하여 구동된다.
Init: 구동되는 경우 신호는 능동발신자에 의해서만 구동된다.
None: 신호는 구동되지 않는다.
Win: 신호는 이기는 장치에 의하여 구동된다.
Targ: 구동되는 경우 신호는 능동수신자에 의해서만 구동된다.

- **ACK:** 발신자에 의하여 리용되며 수신자로부터의 REQ 에 응답한다. ACK 신호는 자료보내기단계기간에 발신자가 모선에 정보를 보낸다는것을 지적하거나 자료받기단계기간에 모선으로부터 자료를 접수했다는것을 지적한다.
- **ATN:** 발신자에 의하여 리용되며 전송에 리용할수 있는 통보문을 가진다는것을 수신자에 알려 주는데 리용된다. 발신자는 수신자가 모선조종을 가정한후에 선택단계나 임의의 시간동안에 이 신호를 낼수 있다.
- **RST**: 모선을 재설정하는데 리용된다.

표 6-4 는 모선신호들과 모선단계들사이의 관계를 보여 준다. 표에는 SCSI-2 에 대한 추가적인 자료 및 기수성선들과 추가적인 REQ/ACK 선들이 있다.

SCSI 시간일치

그림 6-18 은 여러가지 모선단계들과 신호들을 해석하는데 리용되는 대표적인 SCSI 시간일치를 보여 준다. 이 실례는 읽기(Read)지령이며 자료를 수신자로부터 발신자로 전송한다.

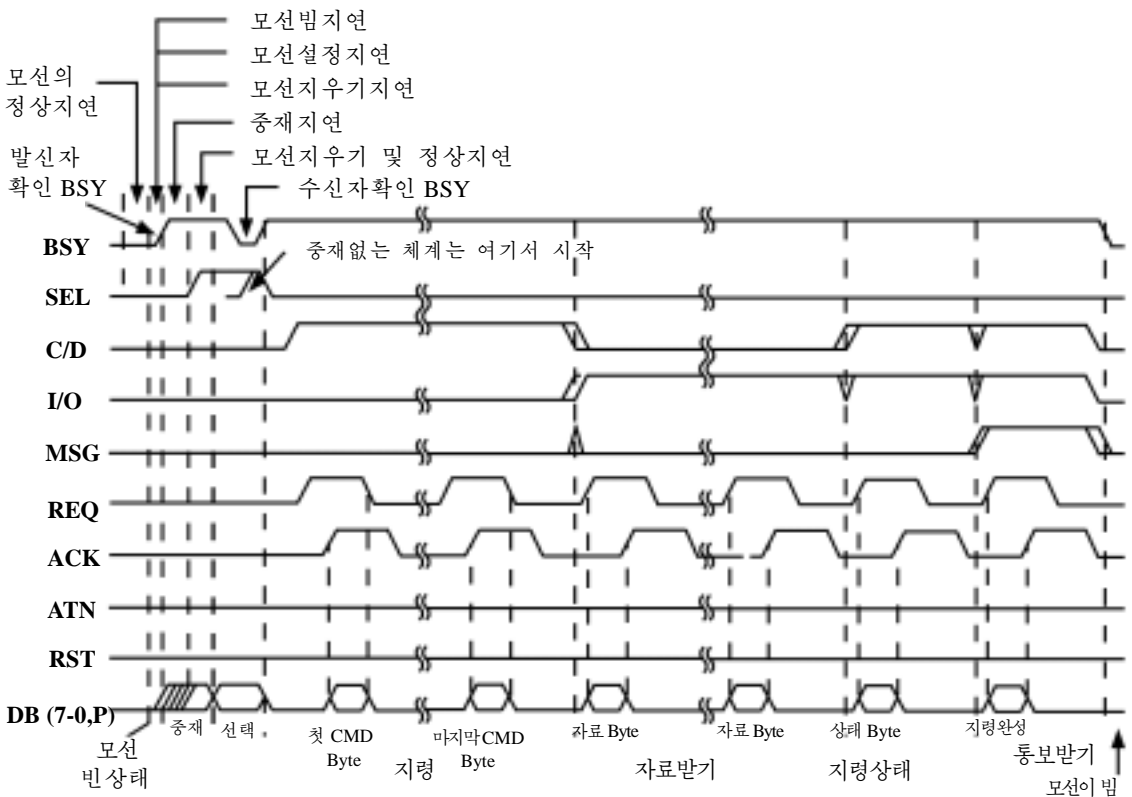



그림 6-18. SCSI 시간선도의 실례

모든 선들에 신호가 없다면 계속하여 모선이 비여 있는 단계를 시작한다. 다음은 중재단계인데 여기서는 하나이상의 장치들이 모선조종권을 쟁취하기 위하여 경쟁한다.

모든 장치들은 BSY 선과 자료선들중의 하나를 점유한다. 8 개의 장치들(한개의 주계산기와 7 개의 다른 장치들)은 0 부터 7 까지의 유일한 ID 를 가진다. 매 장치는 그 자체의 ID 에 대응하는 자료선들중 하나를 점유한다. ID 에는 7 이 가장 높고 0 이 가장 낮은 우선권이 할당된다. 하나이상의 장치가 중재단계기간에 그의 ID 를 점유한다면 가장 높은 우선권을 가진 장치가 이기게 된다. 다른 장치들은 자료선들을 관찰하여 이것을 인식하며 중재를 허용한다.

일단 장치가 중재를 이기면 그것은 곧 발신자로 된다. 그것은 SEL 신호를 내보냄으로써 Selection 단계에 들어 간다. 이 단계기간에 그 자체의 ID 와 수신자 ID 를 모두 두개의 대응하는 자료선들에 보낸다. 그것은 지연된후 BSY 신호를 무시한다. 목적하는 수신자가 SEL 을 내보내며 BSY 와 I/O 가 무시된다는것을 검사하고 그의 ID 를 인식하면 BSY 신호를 보낸다. 발신자는 BSY 를 검사하면 자료모선을 해방하고 SEL 을 무시한다.

다른 수신자는 C/D 선을 점유하여 지령단계에 들어 갔다는것을 지시한다. 즉 이 선은 이 단계 전 기간에 유지되게 된다. 그다음 REQ 를 내보내여 발신자지령의 첫 바이트를 요구한다. 발신자는 지령의 첫 바이트를 자료모선에 놓고 ACK 를 유지한다. 수신자는 그 바이트를 읽은 다음 REQ 를 무시한다. 그다음 발신자는 ACK 를 무시한다. 지령의 첫 바이트는 지령의 몇개의 바이트들이 전송되는가를 가리키는 동작코드를 포함한다. 이 추가적인 바이트들은 매 바이트가 전송되기전과 후에 REQ/ACK 맞잡이동작으로 전송된다.

수신자는 지령을 받고 해석한후 C/D 선을 무시함으로써 자료받기단계기간에 모선을 차지하며(자료모선이 자료를 포함한다는것을 의미한다) I/O 선을 점유한다(전송방향은 수신자로부터 발신자에로 라는것을 의미한다). 수신자는 요구되는 자료의 첫 바이트를 자료모선에 놓고 REQ 선을 보낸다. 발신자는 바이트를 읽은 다음 ACK 선을 내보낸다. 추가적인 자료바이트는 매 바이트가 전송되기전과 전송된후에 REQ/ACK 맞잡이동작으로 전송된다.

요구되는 자료를 모두 전송한후에 수신자는 상태단계에서 모선을 차지하고 상태바이트를 발신자에게 전송하여 전송을 성공적으로 완성했다는것을 지시한다. 이 경우에 C/D 선은 다시 점유되며 I/O 선도 점유를 유지한다. 발신자와 수신자는 상태바이트전송과 같이 REQ/ACK 맞잡이를 리용한다.

끝으로 수신자는 MSG 를 내보내고 지령완료통보를 포함하는 통보문바이트를 보내여 통보받기단계에 모선을 놓는다. 발신자가 이 바이트를 받으면 수신자는 모든 신호들을 해방하여 모선이 비여 있는 단계에 모선을 놓는다.

그림 6-18 에는 동기전송업무의 실례를 보여 주었다. 동기전송에서 REQ/ACK 맞잡이는 전송되는 모든 바이트에 요구된다. SCSI 는 동기전송을 지원하는데 이 동기방식은 보다 적은 조종시간을 요구한다. 동기전송방식은 자료받기 과 자료보내기단계들에서만 리용된다. 그것은 암시적인 전송방식이 비동기이므로 동기방식은 협상되여야 리용될수 있기때문이다. 수신자는 발신자에게 REQ 와 그에 대응하는 ACK 사이에서 허용되는 최소전송주기와 최대편차를 포함하는 동기자료전송요구(Sychronous Data Transfer Request)통보를 보낸다. 발신자는 같은 통보문에 응답하여 그 자체의 최소전송주기와

최대 REQ/ACK 편차를 가리킨다.

일단 이 교환이 일어 나면 동기전송방식은 더 큰 두가지 최소전송주기와 더 적은 두가지 최대 REQ/ACK 편차들로 실현되게 된다. 그 다음 자료전송이 다음과 같이 일어난다. 송신측은 적어도 최소전송주기들로 분할된 자료바이트들을 보내여 REQ 신호를 발생시켜 매 바이트에 알려 준다. 수신측은 즉시 매 수신된 바이트에 ACK 를 응답할 필요는 없으나 편차주기내에서 한 바이트로 응답해야 한다. 송신측은 편차주기내에 대응하는 ACK 들을 받는 한 련속적인 바이트흐름을 보낼수 있다. ACK 가 늦어 지면 송신측은 응답을 받기 위하여 일시 정지되여야 한다.

통보문

통보문들은 SCSI 대면부를 관리할 목적으로 발신자와 수신자사이에서 교환된다. 통보문에는 1byte, 2byte 그리고 3byte 이상의 길이를 가진 확장통보문 등의 세가지 통보문이 있다. 그 실례들은 다음과 같다.

- **지령완료(Command Complete):** 수신자가 발신자에 보내는 통보문으로서 지령이 끝나서 유효상태가 발신자에게 주어 졌다는것을 가리킨다.
- **차단(Disconnect):** 수신자로부터 발신자에게 보내진다. 현재 련결이 끊어 지지만 현재 조작을 완성하기 위하여 후에 재련결이 요구될것이라는것을 통지한다.
- **발신자가 검출한 오유(Initiator Detected Error):** 발신자가 기우성오유가 발생하였다는것을 수신자에게 통지한다. 수신자는 동작을 재개할수 없다.
- **실패(Abort):** 발신자가 현재동작을 무시하도록 수신자에게 보낸다.
- **동기자료전송(Synchronous Data Transfer):** 발신자와 수신자사이에서 교환되며 동기자료전송을 확립한다.

지령

SCSI 규약의 심장부는 지령묶음이다. 발신자는 수신자에게 일부 작용이 가해 지도록 지령을 내보낸다. 지령은 수신자로부터 자료를 회복하기(읽기), 수신자에게 자료를 보내기(쓰기) 또는 특정한 주변장치에 리용되는 다른 동작 등을 동반한다. 모든 경우에 지령실행은 다음의 단계들의 일부 또는 모두 포함한다.

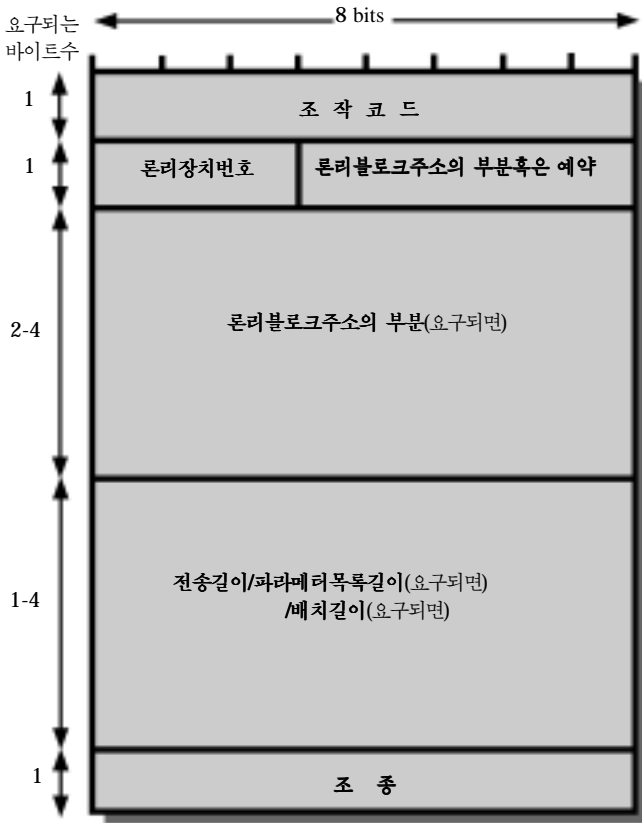
- 수신자는 지령정보를 접수하고 해신한다.
- 자료는 수신자에로 또는 수신자로부터 전송된다.
- 수신자는 상태정보를 발생시키고 되돌린다.

지령은 발신자가 준비한 지령서술자블로크(CDB)에 정의된다. 일단 발신자와 수신자사이에 련결이 이루어 지면 발신자는 CDB 를 자료모선을 통하여 수신자에게 보내여 지령을 실행한다.

그림 6-19 는 CDB 의 일반적인 형식을 보여 준다. 이것은 다음과 같은 마당들로 이루어 진다.

- **조작코드:** 이 코드는 어떤 지령인가를 규정한다. 또한 조작코드는 CDB 의 나머

지형식을 나타낸다.

그림 6-19. SCSI 지령서술자 블로크형식

- **론리장치번호:** 수신자에게 붙은 물리적인 또는 가상적인 장치를 식별한다. 론리장치번호들은 공통조종소자들을 공유하는 여러개의 주변장치들이나 디스크의 여러개의 론리적인 체적들을 주소화하는데 리용될수 있다.
- **론리블로크주소:** 읽기-쓰기동작에서 CDB 는 전송을 위한 론리시작주소를 포함한다.
- **전송길이:** 이 마당은 읽기-쓰기동작에서 전송될 자료의 련속적인 론리블로크수를 규정한다.
- **파라메터목록길이:** 자료보내기단계동안에 보내진 바이트수를 규정한다. 이 마당은 일반적으로 수신자에 보내지는 파라메터들에 리용된다.(례를 들어 방식,진단 또는 로그파라메터들)
- **배치길이:** 발신자가 되돌린 자료에 대하여 배치한 최대 바이트수를 규정한다.
- **조종:** 이 마당은 련결과 기발비트들을 포함한다. 이 비트들은 련결된 지령기구를 조종한다. 만일 련결비트가 설정되면 현재 지령은 모선이 비여 있는 단계에서 끝나지 않지만 대신 지령단계를 시작하여 다음지령과 접촉한다. 만일 기발비트가 설정되면 수신자는 지령이 성과적으로 완성되여 CDB 에서와 같은 기발값을 되돌릴때 련결지령완료통보문을 보낸다. 일반적으로 련결지령들사이에 발신자에 의하여 새치기가 발생하도록 기발값 1 을 리용한다.

SCSI 는 넓은 령역의 지령형태들을 정의한다. 이 대부분은 특수한 종류의 장치에 대하여 고유하다. SCSI-2 는 다음과 같은 장치형태들에 대한 지령을 포함한다.

- 직접접근장치
- 련속접근장치
- 인쇄기

- 처리장치
- 한번만 쓰기장치
- CD-ROM
- 스캐너
- 빛기억장치
- 매체변환장치
- 통신장치

또한 모든 장치형태들에 리용되는 17 개의 지령묶음이 있다. 물론 다음의 4 개는 모든 장치들에서 필수적인것으로서 반드시 실현되여야 한다.

- **물음(Inquiry):** 수신자와 그에 접속된 주변장치들의 파라메터들을 발신자에게 보낼것을 요구한다.
- **수감요구(Request Sense):** 수신자가 발신자에게 수감자료를 전송할것을 요구한다. 수값자료에는 오유조건(례:종이없음), 위치정보(례: 매체끝), 론리상태정보(례: 현재지령이 filemark 와 접촉하였다.) 등이 있다.
- **진단보내기(Send Diagnostic):** 수신자는 그 자체에 대하여 접속된 주변장치들에 대하여 또는 이 둘에 대하여 진단시험을 수행할것을 요구한다.
- **시험장치준비(Test unit ready):** 론리장치가 준비되였는가를 검사하는 수단을 준다.

SCSI 의 지령저장고는 이 규정의 주요우점이다. 그것은 주체계에서의 I/O 쏘프트웨어를 작성하는 과제를 간단화하는 개인용콤퓨터와 워크스테이션에 접속된 가장 공통적인 장치들을 취급하는 표준방법을 제공한다.

4. FireWire 직렬모선

처리장치의 속도가 100MHz에 이르고 대용량기억장치들이 수Gbit를 보관할수 있게 되면서 개인용콤퓨터, 워크스테이션, 봉사기들에 대한 I/O 요구조건들도 높아 지게 되였다. 대형콤퓨터와 초고속콤퓨터체계에서 개발된 높은 속도 I/O 통로기술을 이 작은 체계에 리용하기에는 아직 값이 너무 비싸고 부피가 크다. 따라서 SCSI 와 다른 작은 체계 I/O 대면부들에 리용할수 있는 고속대치방도를 개발하는데 큰 관심을 가지게 되였다. 그 결과 IEEE 표준 1394 가 나오게 되였으며 이것이 바로 일반적으로 FireWire 로 알려 진 고성능직렬모선이다.

FireWire 는 SCSI 와 다른 I/O 대면부들보다 많은 우점을 가진다. 즉 속도가 매우 높고 가격이 눅으며 그 실현이 쉽다. 사실상 FireWire는 콤퓨터체계에서뿐아니라 수자식카메라, VCR, TV 와 같은 소비자용전자제품들에서도 인기가 있다. 이 제품들에서 FireWire 는 수자화된 원천으로부터 점차적으로 오는 비데오화상들을 전달하는데 리용된다.

FireWire 대면부의 한가지 우점은 그것이 병렬전송이 아니라 직렬전송(한번에 한 비트)을 리용한다는것이다. SCSI 와 같은 병렬대면부들은 심선들이 많을수록 더 굵어지

고 더 비싸다. 그리고 케블들이 굵을수록 구부러지거나 꺾이는 부위들이 많아 지며 더 비싼 접속부들이 요구된다. 더 많은 심선들을 가진 케블은 심선들사이의 전기적간섭을 막기 위한 차페를 요구한다. 또한 병렬대면부에서 선들사이의 동기화가 요구되며 케블길이가 증가함에 따라 더 나쁜 현상들이 나타난다.

더우기 콤퓨터는 계산능력과 I/O 요구가 증가하는 반면에 물리적으로는 더 작아 지고 있다 .손바닥콤퓨터와 주머니크기콤퓨터들은 접속기들에 대하여서는 거의나 공간을 소비하지 않지만 화상과 비데오를 조종하는데는 높은 자료속도를 요구한다.

FireWire 의 목적은 단일포구를 통하여 여러 장치들을 조종할수 있는 간단한 접속기를 가진 단일한 I/O 대면부를 제공하는것이다. 그리하여 마우스, 레이자인쇄기, SCSI, 외부디스크구동기, 스피카, 론리령역망접속들이 이 단일접속기로 교체될수 있다. 이 접속기는 Nintendo Gameboy 에서 리용된것에서 착상되였다. 그것은 아주 편리하여 사용자가 기계뒤에 가서 보지 않고도 그것을 끼울수 있다.

5. FireWire 구성

FireWire 는 63 개까지의 장치들을 단일포구에 련결할수 있는 련쇄구성을 리용한다. 더우기 1022 까지의 FireWire 모선들이 다리를 리용하여 호상련결될수 있으므로 체계가 요구하는것만큼의 많은 주변장치들을 지원할수 있게 한다.

FireWire 는 콤퓨터의 전원을 끄거나 체계를 재구성하지 않고 주변장치들을 련결하거나 차단할수 있게 하는 작업중 접속(hot plugging)이라고 하는 기능을 제공한다. 또한 FireWire 는 자동구성을 제공한다. 즉 장치 ID 설정이나 장치들의 상대적위치와 련관시키는것을 수동적으로 하지 않는다. 그림 6-20 은 FireWire 구성을 SCSI 와 비교한다. SCSI 에서 모선의 두 끌에는 종단기가 있으며 매 장치에는 구성부분으로서 유일주소가

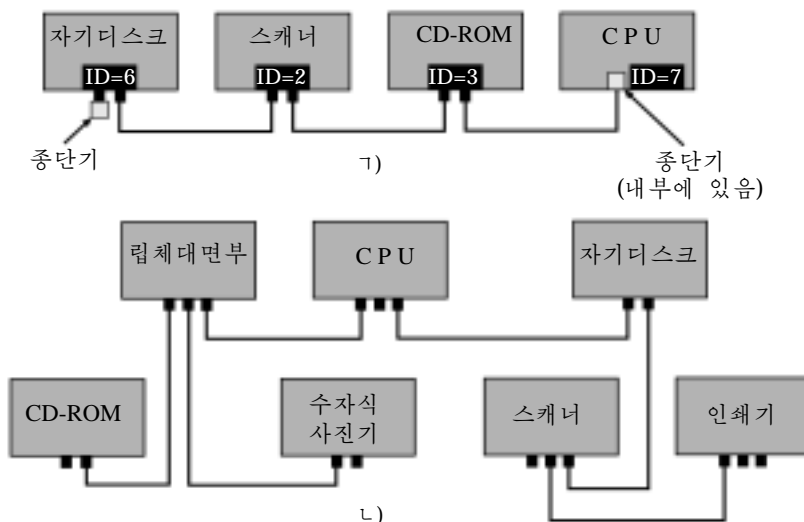



그림 6-20. SCSI 와 고속직렬포구의 비교
ㄱ-SCSI 구성실례, ㄴ-고속직렬포구구성실례

할당된다.

FireWire 에는 종단이 없으며 체계는 자동적으로 주소를 할당하기 위한 구성기능을 수행한다. 또한 FireWire 모선은 엄밀한 련쇄가 아니여도 된다. 오히려 나무구조구성이 가능하다.

FireWire 규격의 주요특징은 주체계가 직렬모선을 통하여 주변장치들과 호상작용하는 방법들을 표준화하는 세 층으로 된 규약묶음을 규정한것이다. 그림 6-21 은 이 탄창을 보여 준다. 탄창의 세 층은 다음과 같다.

- **물리층:** FireWire 에서 허용되는 전송매체와 전기적 및 신호적특징들을 각각 정의한다.
- **련결층:** 자료전송을 파케트로 묘사한다.
- **업무층:** 응용으로부터 FireWire 의 보다 낮은 층세부를 은페시키는 요구-응답 규약을 정의한다.

물리층

FireWire 의 물리층은 각이한 물리적속성들과 자료전송속성들과 함께 여러가지 다른 전송매체와 그 접속기들을 규정한다. 자료속도는 25~400Mbps 로 정의된다. 물리층은 2 진자료를 여러가지 물리적매체에 알맞는 전기신호로 변환한다. 이 층은 한번에 단 하나의 장치만이 자료를 전송할것을 담보하는 중재봉사를 제공한다.

FireWire 는 두가지 형태의 중재를 제공한다. 가장 단순한 형태는 초기에 언급된바와 같이 FireWire 모선우에서 마디들의 나무-구조할당에 기초한다. 이 구조의 특수경우는 선형련쇄이다. 물리층은 한 마디는 나무의 뿌리로서 설계되고 다른 마디들은 부모/자

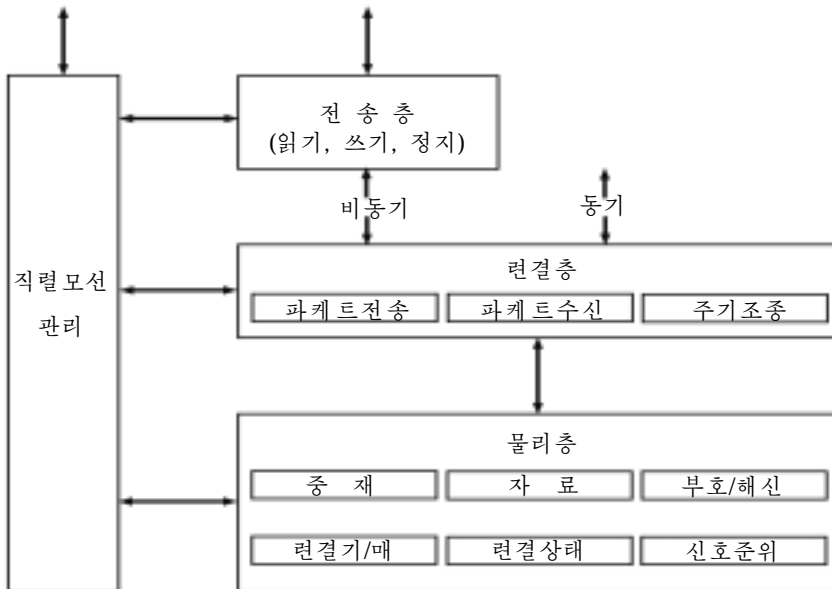



그림 6-21. 고속직렬포구규약블로크도

식관계로 조직화되여 나무배치를 이루도록 자체로 모든 접속된 장치들을 구성하는 론리를 포함한다. 일단 이 구성이 이루어 지면 뿌리마디는 중심중재기로서 작용하여 처음에 온것이 처음에 봉사되는 방식으로 모선접근에 대한 요구를 처리한다. 동시에 일어 나는 요구에 대하여 가장 높은 자연우선권을 가진 마디가 접근된다. 자연우선권은 어느 경쟁마디가 뿌리에 제일 가까운가, 뿌리로부터 같은 거리에 있는가에 의하여 결정되며 더 낮은 ID 번호를 가진다.

앞에서 언급된 중재방법은 두개의 추가적인 기능들 즉 공평중재와 긴급중재에 의하여 실현된다. 공평중재에서 모선우에서의 시간은 공평성간격으로 조직된다. 간격의 초기에 매 마디는 중재, 허가기발을 설정한다. 그 시간간격동안 매 마디는 모선접근을 얻기위해 경쟁한다. 일단 마디가 모선접근을 얻으면 중재-허가기발을 재설정하고 이 시간간격동안에 공평접근에 대해서는 다시 경쟁을 하지 않는다. 이 구성은 중재를 보다 쉽게 하며 하나이상의 높은 우선도를 가진 장치들이 모선을 독점하는것을 막는다.

공평성구성외에도 일부 장치들은 긴급 우선도를 가지도록 구성할수 있다. 그러나 마디들은 공평성간격동안에 모선을 여러번 조종할수 있는 기회를 얻을수 있다. 본질상 계수기는 우선도가 높은 마디들이 리용되는 모선시간의 75%를 조종하도록 모든 우선도가 높은 마디에서 리용된다. 긴급하지 않게 전송되는 매 파케트에서 세개의 파케트들은 긴급하게 전송될수 있다.

련결층

련결층은 자료전송을 파케트의 형태로 진행한다. 두가지 형태의 전송을 지원한다.

- **비동기**: 각이한 크기의 자료와 여러 바이트의 업무층정보가 명시적인 주소에 파케트로서 전송되며 응답이 되돌려 진다.
- **동기**: 각이한 크기의 자료는 고정길이파케트렬로 변환되여 일정한 간격으로 전송된다. 이 전송형태의 주소는 간단화된 주소를 리용하며 아무 응답도 없다.

고정된 자료속도를 요구하지 않는 자료는 비동기적으로 전송된다. 공평중재와 긴급중재방식들은 다 비동기전송에 리용할수 있다. 암시적방법은 공평중재이다. 본질적으로 모선능력을 요구하지만 심한 지연을 요구하는 장치들은 긴급중재방법을 리용한다. 례를 들어 고속실시간자료수집마디는 자료완충기가 절반이상 찼을 때 긴급중재를 리용할수 있다.

그림 6-22 ㄱ는 대표적인 비동기업무를 보여 준다. 단일한 파케트를 전달하는 처리를 보조작용이라고 한다. 보조작용은 5개의 시간주기로 이루어 진다.

- **중재순서**: 이것은 어떤 장치에 모선조종을 주도록 요구한 신호들의 교환이다.
- **파케트전송**: 모든 파케트는 원천 ID 와 목적 ID 를 포함하는 머리부를 가지고 있다. 머리부에는 또한 파케트형태정보, 순환여유검사(CRC), 특수 파케트형태에 대한 파라메터정보가 있다. 파케트는 또한 사용자자료와 또 다른 CRC 로 이루어 진 자료블로크를 포함한다.
- **응답간격**: 이것은 목적지가 파케트를 수신하고 해신하여 응답을 발생시키는데 필요한 시간이다.

- **응답:** 파케트의 수신자는 수신자가 취한 작용을 가리키는 코드를 가지는 응답파케트를 되돌린다.
- **보조작용간격:** 이것은 모선상의 다른 마디들이 응답파케트들이 전송되기전에 중재를 시작하지 않는다는것을 담보하는 강제적인 빈 주기이다.

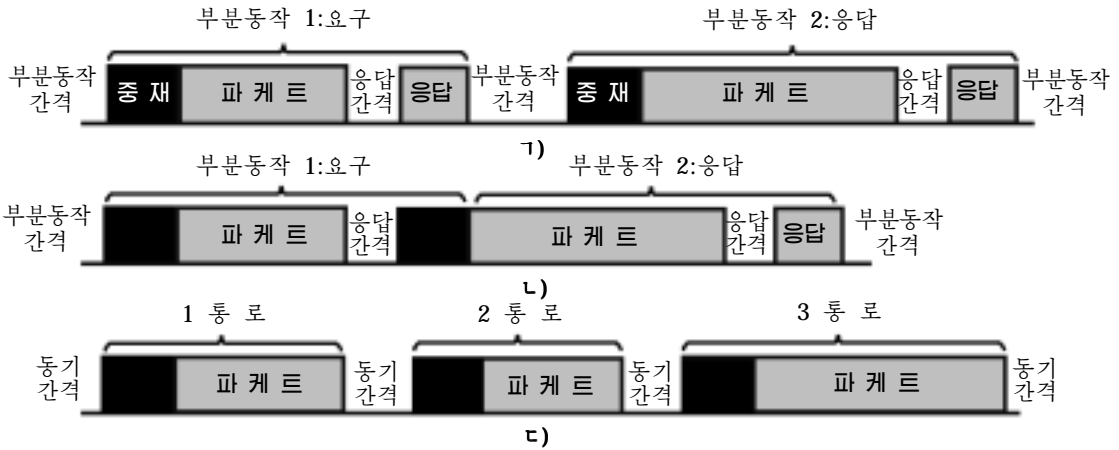



그림 6-22. 고속직렬포구부분동작
ㄱ-비동기부분동작실례, ㄴ-련결된 비동기부분동작, ㄷ-동기부분동작실례

응답이 전송될 때에는 응답마디가 모선의 조종하에 있게 된다. 그러므로 교환이 두 마디사이 요구/응답호상작용이라면 응답마디는 직접적으로 중재공정을 거치지 않고 응답파케트를 전송할수 있다(그림 6-22 ㄴ).

수자음성이나 비데오와 같이 자료를 규칙적으로 발생하거나 소비하는 장치에서는 동기접근이 보장된다. 이 방법은 자료가 담보된 자료속도로 규정된 대기시간내에 전달될 수 있다는것을 담보한다.

동기 및 비동기자료원천의 혼합통신량부하를 모두 처리하기 위하여 하나의 마디가 **기본순환주기시동자**로 지적된다. 이 기본순환주기시동자는 주기적으로 순환시작파케트를 보낸다. 이것은 다른 모든 마디들에 동기주기가 시작되였다는것을 통지한다. 이 순환주기기간에는 오직 동기파케트만이 보내 질수 있다(그림 6-22 ㄷ). 매 동기자료원천은 직접적으로 모선접근을 진행한다. 이긴 마디는 즉시 파케트를 전송한다. 이 파케트는 응답이 없으며 따라서 다른 동기자료원천은 앞의 동기파케트가 전송된 후에 모선에 대한 직접중재를 진행한다. 그 결과 하나의 파케트전송과 다음 파케트중재주기까지의 사이에 작은 간격이 생기며 이것은 모선우에서의 지연으로 결정된다. 이 지연을 동기간격이라고 하며 이것은 보조작용간격보다 작다.

모든 동기원천이 전송된 후에 모선은 보조작용간격이 나타나기에는 충분한 긴 간격을 남긴다. 이것은 모선접근을 완성하기 위한 비동기원천의 신호로 된다. 이 동기원천은 다음주기가 올 때까지 모선을 리용할수 있다.

동기파케트들은 동기자료를 교환하려는 두 마디사이 회화에 의하여 이전에 할당된 8bit 통로번호로 표식이 붙게 된다. 머리부는 비동기파케트의 머리부보다 더 짧으면서도 자료길이마당과 머리부 CRC 를 포함한다.

참고문헌과 Web 싸이트

8255A 와 82C55A 를 비롯한 Intel 계렬 I/O 모듈과 기본구성에 대하여서는 [BRAY97]에서 잘 고찰하고 있다.

SCSI 에 대한 좋은 개론을 주는 책은 [SCHM97]와 [NCR90]의 두 책이며 [ANDE98]에서는 FireWire 에 대하여 구체적으로 고찰하고 있다.

ANDE98 AnDerson, D. FireWire System Architecture. Reading, MA: Addison-Wesley, 1998

BREY97 Brey, B. The Intel Microprocessors:8086/8066, 80186.80188, 80286, 80386, 80486, Pentium, and Pentium Processor. Upper Saddle River, NJ: Prentice Hall, 1997.

NCR90 NCR Corp. SCSI: Understanding the Small Computer System Interface. Englewood Cliffs, NJ: Prentice Hall, 1990.

SCHM97 Schmidt, F. The SCSI Bus and IDE Interface. Reading, MA: Addison-Wesley, 1997.

Web 싸이트:

- **T10 Home Page:** T10 은 정보기술표준국가위원회의 기술위원회 (Technical Committee of the National Committee on Information Technology Standards)이며 보다 낮은 준위대면부에 대응한다. 이 위원회의 기본작업은 작은 콤퓨터체계대면부 (SCSI)이다.
- **SCSI Trade Association:** 기술정보와 통신회선제공회사 간판 (Vendor Pointer)을 포함한다.
- **B94 Trade Association:** 기술정보와 FireWire 에 대한 통신회선제공회사 간판을 포함한다.

련습문제

1. 제 6 장 제 3 절에서는 분리된 I/O 와 비교하여 기억기배치식 I/O 의 한가지 우점과 결함을 서술하였다. 우점과 결함을 두가지 이상 말해 보시오.
2. DMA 장치를 포함한 모든 가상체계에서 주기억기에 대한 DMA 접근은 주기억기에 대한 CPU 접근보다 더 높은 우선권을 가진다. 왜 그런가?

3. 문제 5,6에 서술된 디스크체계를 보자. 여기서 디스크회전속도가 360r/min이라고 하자. 처리장치가 바이트당 하나의 새치기를 가지는 새치기구동 I/O를 써서 디스크로부터 한개 분구를 읽는다. 매 새치기를 처리하는데 2.5μs가 걸린다면 처리장치가 I/O를 구동하는데 몇 %의 시간이 들겠는가?

4. DMA를 써서 문제 3을 다시 해보시오. 분구당 한개 새치기를 가정하시오.

5. DMA 장치는 9600bps로 자료를 전송하는 장치로부터 순환주기훔침을 리용하여 기억기로 문자를 전송하고 있다. 처리장치는 초당 100만개 지령의 속도(1MIPS)로 지령을 꺼낸다. DMA의 작용으로 처리장치는 얼마나 떠지는가?

6. 32bit콤퓨터에는 두개의 선택기통로와 다중화장치통로가 있다. 매 선택기통로는 두개의 자기디스크와 자기테프장치를 지원한다. 다중화장치통로는 두개의 행인쇄기, 두개의 카드읽기장치, 그에 련결된 10개의 VDT 말단을 가진다. 이제 전송속도가 다음과 같다고 가정하자.

   디스크구동기: 800kbyte/s

   자기테프구동기: 200kbyte/s

   행인쇄기: 6.6kbyte/s

   카드읽기장치: 1.2kbyte/s

   VDT: 1kbyte/s

   이 체계에서 최대 총 I/O전송속도를 평가하시오.

7. 콤퓨터가 하나의 처리장치와 한개 단어의 너비로 모선을 공유하여 주기억기 M에 련결된 I/O모듈 D로 이루어 져 있다고 하자. 처리장치는 최대 초당 10^6개의 지령을 실행할수 있다. 평균 한개 지령은 5개의 기계주기를 차지하는데 그중 세개는 기억기모선을 리용한다. 기억기의 읽기쓰기조작은 한개 기계주기를 사용한다. 처리장치는 련속적으로 《배경》프로그람을 집행하는데 이 프로그람은 95%가 지령실행이고 나머지가 I/O지령이라고 하자. 하나의 처리장치주기는 하나의 모선주기를 요구한다고 가정하자. 이제 I/O 모듈이 아주 큰 자료블로크를 M과 D 사이에 전송한다고 하자.

   ㄱ. 만일 프로그람식 I/O가 리용되고 매개 한개 단어 I/O 전송이 두개 지령을 실행할것을 처리장치에 요구한다면 D를 통하여 가능한 I/O 자료전송속도(초당 단어)의 최대값을 평가해 보시오.

   ㄴ. DMA가 리용된다면 이와 같은 속도를 평가해 보시오.

8. 자료원천은 7bit ASCII 문자이고 여기에 기우성비트를 첨가하자. 다음의 경우에 R-bps행 이상의 최대유효자료속도에 대한 식을 유도하시오.

   ㄱ. 1.5단위정지비트를 가진 비동기전송

   ㄴ. 48개의 조종비트와 128개의 정보비트로 구성된 틀을 가진 비트동기전송

ㄷ. ㄴ와 같은데 1024bit의 정보마당이 있다.
ㄹ. 틀당 9개의 조종문자를 가지고 있고 16개의 정보문자가 있는 문자동기
ㅁ. ㄹ와 같은데 128개의 정보문자를 가지고 있다.

9. 다음문제는 [ECKE90]에서 제안된 I/O원리에 기초한다(그림 6-23). 두 소년들이 높은 울타리의 서로 다른 쪽에서 놀고 있다. 그중 한 소년의 이름은 사과공급자로서 그는 먹음직한 사과가 주렁주렁 달린 아름다운 사과나무를 가지고 있는데 다른쪽 소년이 요구할 때마다 사과를 준다. 다른 소년은 사과먹새기라고 하는데 사과를 먹기만 하고 아무것도 하지 않는다. 그러나 그는 사과를 고정된 속도로 먹어야 한다. 정해 진 속도보다 빨리 먹으면 그는 앓게 된다. 천천히 먹으면 배고파 한다. 소년은 말할수도 없으며 따라서 문제는 사과공급자로부터 사과먹새기로 정해 진 속도로 사과를 보내주는것이다.

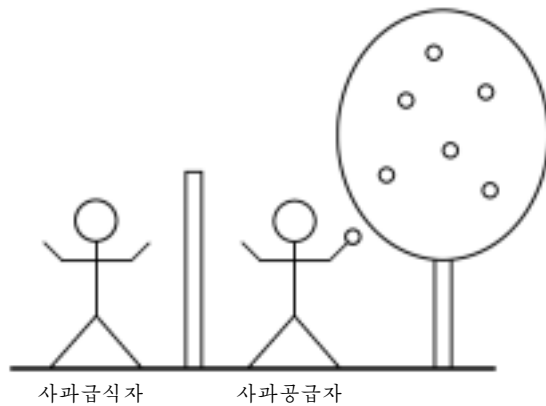



그림 6-23. 사과문제

ㄱ. 울타리짝대기에 경보시계가 있으며 그 경보시계가 여러가지 경보를 낼수 있다고 하자. 경보시계를 리용하여 이 문제를 어떻게 풀수 있는가? 풀이를 위한 시간선도를 그려 보시오.
ㄴ. 경보시계가 없다고 하자. 대신 사과먹새기가 사과를 요구할 때마다 흔들수 있는 기발을 가지고 있다고 하자. 새로운 풀이를 구하시오. 사과공급자도 기발을 가지면 좋겠는가? 만일 그렇다면 이것은 부정확한 풀이로 된다. 이 방법의 결함을 론의해 보시오.
ㄷ. 이제 기발대신 문자렬의 긴 쪼각을 생각하자. 문자렬을 써서 ㄴ에서 보다 좋은 풀이를 찾으시오.

10. 16bit와 8bit극소형처리장치가 체계모선과 결합되여 있다고 하자. 다음과 같은것들이 상세히 주어져 있다고 하자.
① 모든 극소형처리장치는 임의의 형태의 자료전송에 필요한 장치적특징을 가지고

있다.

② 모든 처리장치는 16bit 주소모선을 가진다.

③ 매 용량이 64KB 은 두개의 기억기기판이 모선과 결합되여 있다. 설계가는 될수록 큰 공유기억기를 리용하려고 한다.

④ 체계모선은 최대로 4 개의 새치기선과 DMA 선을 가진다.

필요한 임의의 다른 가정을 할수도 있다.

ㄱ. 선의 종류와 번호에 의해서 체계모선을 지정하시오.

ㄴ. 앞에서 서술한 장치가 어떻게 체계모선에 결합되는가를 설명하시오.

참고: [ALEX93]

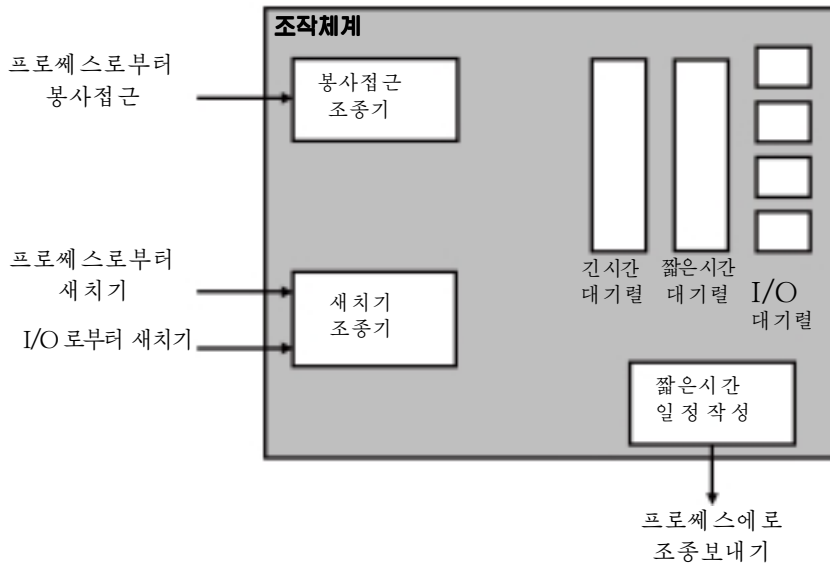
11. SCSI 모선우에서 매 I/O 모듈는 가장 빠른 속도인 돌발전송속도를 결정하기 위해 주콤퓨터와 협상한다. 최대돌발전송속도가 주콤퓨터에서 20MB/s 라고 하자. 모든 장치가 적당한 완충기를 가지면 모선시간내의 경쟁에서도 전송을 계속 유지한다고 하자.

ㄱ. 500Kbyte/s 의 전송속도와 4Mbyte/s 의 전송속도로 테프구동기가 SCSI 에 결합되여 있다고 하자. 같은 모선에 디스크구동기를 각각 6Mbyte/s 의 전송속도와 20Mbyte/s 의 돌발속도로 접속할것을 원한다고 하자. 모든 장치를 최대속도로 동시에 기동시키자면 모선에 최대 몇대까지 접속시킬 수 있는가?

풀이방향: 모든 자료를 전송하는데 필요한 매 장치의 시간비률을 결정하시오(주의: 1K를 1024가 아니라 1000으로, 1M를 1,048,576이 아니라 1,000,000 으로 하시오. 이 근사는 이 문제의 목적과 아주 가깝다. 전송속도는 보통 10 진수에 기초한 수를 리용하여 표시되지만 완충기크기와 전송크기와 같은 수들은 보통 2 진수에 기초한 수들을 리용하여 표시된다.).

ㄴ. 이제 이 모형을 좀 더 고찰해 보자. 테프구동기가 4ms 의 모선간접소비시간을 요구하며 최대 전송크기가 64Kbyte 라고 하자. 디스크구동기 역시 전송할 때마다 4ms의 모선간접소비시간이 들며 그때 전송마다 256Kbyte 의 량을 전송한다. 매 장치당 모선의 점유률과 총체적인 점유률을 재평가하시오. 모선이 ㄱ에서의 당신의 대답에서 규정된 장치수에 알맞는다고 생각하는가?

제 7 장. 조작체계의 지원

◆◆

◆ 조작체계는 처리장치에서 프로그람들의 집행을 조종하고 처리장치자원들을 관리하는 쏘프트웨어이다. 프로쎄스일정작성과 기억기관리를 포함하는 조작체계의 여러가지 기능들은 처리장치하드웨어가 조작체계를 지원하기 위한 능력들을 가진다면 더 능률적으로 빨리 수행될수 있다. 실제적으로 모든 처리장치들은 가상기억관리하드웨어와 프로쎄스관리하드웨어를 포함하여 크고 작은 범위에서 이와 같은 능력을 가진다. 하드웨어는 기본자원관리과제들을 수행하기 위한 구성요소들과 같은 특수등록기들과 완충기들을 가진다.

◆ 조작체계의 가장 중요한 기능의 하나는 프로쎄스들 혹은 과제들에 대한 일정작성이다. 조작체계는 프로쎄스가 어떤 주어 진 시간에 집행되여야 한다는것을 결정한다. 일반적으로 하드웨어는 때때로 조작체계가 여러 프로쎄스들사이에 공정하게 처리시간을 분할하도록 새로운 일정작성을 진행할수 있도록 프로쎄스집행을 중단할수 있다.

◆ 조작체계의 다른 중요한 기능은 기억기관리이다. 최근의 조작체계들은 가상기억기기능을 가지고 있는데 두가지 우점이 있다. 첫째로, 프로쎄스는 프로그람의 명령과 자료가 동시에 주기억기에 상주하고 있지 않아도 주기억기상에서 집행할수 있다는것이다. 둘째로, 프로그람집행에 필요한 기억공간은 체계에 따르는 실제의 기억공간보다 훨씬 더 크다. 비록 기억기관리가 쏘프트웨어에 의하여 수행된다할지라도 조작체계는 장치적으로 페지화와 토막화를 수행하는것을 비롯한 처리장치안에서의 하드웨어적인 지원에 의거한다.

◆◆

이 책에서는 콤퓨터하드웨어에 대하여 주로 고찰하고 있지만 쏘프트웨어의 한 부분인 조작체계에 대해서도 필요한 만큼 서술하고 있다. 조작체계는 콤퓨터자원을 관리하여 프로그람작성자에게 봉사를 제공하며 다른 프로그람의 집행순서를 작성해 주는 프로그람이다. 조작체계를 리해함에 있어서 기본은 CPU 가 콤퓨터체계를 조종하는 기구라는것을 인식하는것이다. 이 책에서는 특히 새치기효과와 기억기의 계층관리에 대해서 구체적으로 설명하였다.

이 장에서는 콤퓨터조직과 구성방식의 연구에서 제일 중요한 조작체계의 두가지 기능인 일정작성과 기억관리를 주로 취급하였다.

제 1 절. 일반개념

1. 조작체계목적과 기능

조작체계는 응용프로그람의 집행을 조종하고 콤퓨터사용자와 콤퓨터하드웨어사이의 대면부로서의 기능을 수행하는 프로그람이다. 그 필요성은 두가지로 볼수 있다.

- 편리성: 조작체계는 콤퓨터를 더 편리하게 사용하게 한다.
- 효과성: 조작체계는 콤퓨터체계자원들을 더 효과적인 방법으로 리용하게 한다.

조작체계의 이 두가지 측면을 차례로 설명하기로 하자.

사용자/콤퓨터대면부로서의 조작체계

그림 7-1 에서 보여 준것처럼 응용프로그람을 사용자에게 제공하는데 리용되는 하드웨어와 쏘프트웨어는 계층형식으로 고찰할수 있다. 이러한 응용프로그람들의 사용자는 일반적으로 콤퓨터기본방식과는 무관계하다. 따라서 사용자는 응용프로그람의 견지에서 콤퓨터체계를 고찰한다. 그 응용프로그람은 프로그람작성언어로 표현되고 응용프로그람작성자에 의하여 개발된다. 만일 콤퓨터하드웨어를 직접적으로 조종하는 기능을

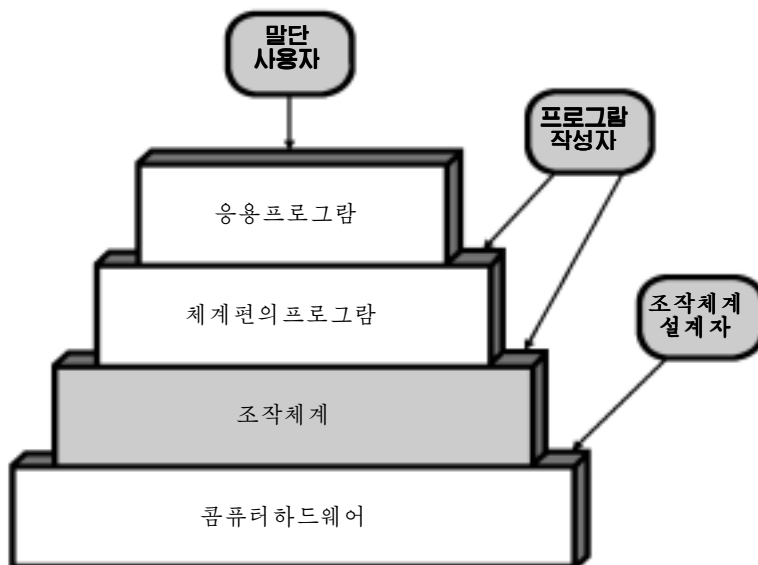



그림 7-1. 콤퓨터체계의 계층과 고찰

가진 처리장치명령들의 묶음으로서 응용프로그람을 개발한다고 하면 아주 복잡한 과제에 부닥치게 될것이다. 이 과제를 쉽게 하기 위하여 체계프로그람묶음이 제공된다. 이 프로그람들의 일부를 편의프로그람이라고 한다. 이 도구들은 프로그람개발과 파일관리, I/O 장치조종을 돕는 기능들을 가진다. 프로그람작성자는 이 기능을 응용프로그람개발에 리용하며 한편 응용프로그람이 집행되는 동안에는 일정한 기능을 수행하기 위하여 편의프로그람들을 요구한다. 가장 중요한 체계프로그람은 조작체계이다. 조작체계는 프로그람작성자를 대신하여 하드웨어의 구체적인 서술을 담당하며 프로그람작성자에게 체계를 리용하는데 도움을 주는 대면부를 제공한다. 이것은 프로그람작성자와 응용프로그람이 이 기능의 봉사를 접근하고 리용하는데서 중매적인 기능을 수행한다. 조작체계가 제공하는 봉사는 대표적으로 다음과 같이 나누어 볼수 있다.

- **프로그람작성:** 조작체계는 프로그람작성자의 프로그람개발을 돕기 위한 편집프로그람, 오유수정프로그람과 같은 여러가지 편리와 봉사를 제공한다. 일반적으로 이 봉사들은 조작체계의 실제적인 부분이 아니지만 조작체계에 의하여 접근될수 있는 편의프로그람의 형태로 존재한다.
- **프로그람집행:** 많은 과제들은 프로그람집행으로 수행될것을 요구한다. 명령들과 자료는 주기억기에 넣어 져야 하고 입출력장치들과 파일들은 초기화되여야 하며 다른 자원들이 준비되여야 한다. 조작체계는 사용자들에 대하여 이 모든것을 조종한다.
- **입출력장치에 대한 접근:** 매 입출력장치들은 조작을 위하여 그자체의 고유한 명령묶음이나 조종신호들을 요구한다. 조작체계는 프로그람작성자가 간단한 읽기와 쓰기로서 생각할수 있게 세부적인 처리를 진행한다.
- **파일에 대한 조종접근:** 파일의 경우에 조종은 입출력장치의 본질(디스크구동기, 테프구동기)뿐아니라 기억매체의 파일형식에 대하여 인식하여야 한다. 또한 조작체계는 세부에 대하여 조사한다. 더우기 다중-동시사용자를 가지는 체계인 경우에 조작체계는 파일에 대한 접근을 조종하는 보호기구를 제공할수 있다.
- **체계접근:** 공유 혹은 공동체계인 경우에 조작체계는 전반적인 체계와 지적된 체계자원들에 대한 접근을 조종한다. 이 접근기능은 권한이 없는 사용자로부터의 자원들과 자료보호를 제공하여야 하며 자원충돌에 대한 경쟁을 해결하여야 한다.
- **오유검출과 응답:** 콤퓨터체계가 작업하는 동안 여러가지 오유가 발생할수 있다. 이것들은 기억기오유 혹은 장치결함, 불완전한 기능들과 같은 내적 또는 외적하드웨어오유들과 산수연산의 자리넘침, 금지된 기억주소를 접근하려는 시도, 응용프로그람의 요구를 만족시키는 기능이 조작체계에 없는것과 같은 쏘프트웨어의 여러가지 오유들을 포함한다. 매 경우에 조작체계는 응용프로그람집행에서 자그마한 충격으로 인한 오유조건들을 제거할수 있어야 한다. 이것은 오유를 일으키는 프로그람을 끝내는것으로부터 조작을 재시험하고 응용프로그람의 오유를 간단히 통보하는것까지의 범위에 속한다.
- **평가:** 훌륭한 조작체계는 여러가지 자원들과 응답시간과 같은 감시프로그람성능 파라메터들에 대한 사용통계들을 수집할수 있다. 임의의 체계에서 이 정보는 앞으로의 환경에 대한 요구를 기대하며 성능을 개선하기 위하여 체계를 개조하는데 쓸모 있다. 다중사용자체계에서 이 정보는 선전목적에 사용될수 있다.

자원관리자로서의 조작체계

콤퓨터는 자료의 전송, 보관 그리고 처리와 이러한 기능들의 조종을 위한 자원들

의 묶음이다. 조작체계는 이 자원들을 관리한다.

자료의 전송, 보관 그리고 처리를 조종하는것이 조작체계라고 말할수 있는가? 하나의 관점으로부터 대답은 긍정적이다. 즉 콤퓨터의 자원들을 관리하기때문에 조작체계는 콤퓨터기본기능들의 조종으로 된다. 그러나 이 조종은 기묘한 방법으로 진행된다. 보통 조종기구는 조종되는 외부적인것 혹은 적어도 조종되기 위하여 명백하게 갈라 진 부분으로 생각한다(실례로 존재하는 가열체계는 열발생기와 열분배장치들로 완전히 구별된 자동온도조절장치에 의하여 조종된다.). 이것은 조종기구가 다음의 두가지 례외적인 측면을 가지는 조작체계와는 다르다.

- 조작체계는 보통 콤퓨터쏘프트웨어와 같은 방법으로 작용한다. 즉 처리장치에 의하여 실현되는 프로그람이다.
- 조작체계는 자주 조종을 포기하며 조종을 회복하기 위하여 처리장치에 의거한다.

사실상 조작체계는 콤퓨터프로그람에 불과하다. 다른 콤퓨터프로그람과 같이 조작체계는 처리장치에 대하여 명령들을 제공한다. 기본차이는 프로그람의 목적에 있다. 조작체계는 다른 체계자원의 리용과 다른 프로그람집행의 동기화로 처리장치를 직접 관리한다. 그러나 처리장치는 일부 프로그람의 집행을 위하여 조작체계프로그람의 집행을 중지하고 다른 프로그람을 집행한다. 따라서 조작체계는 일부 《필요한》작업집행을 위하여 처리장치에 대한 조종을 포기하고 처리장치가 다음의 부분작업을 진행하도록 준비하는데 필요한 충분한 조종을 다시 진행한다. 이 모든것에 대한 기구는 아래로 내려 가면서 명백히 하려고 한다.

그림 7-2 는 조작체계에 의하여 관리되는 주요자원들을 보여 준다. 조작체계부분은

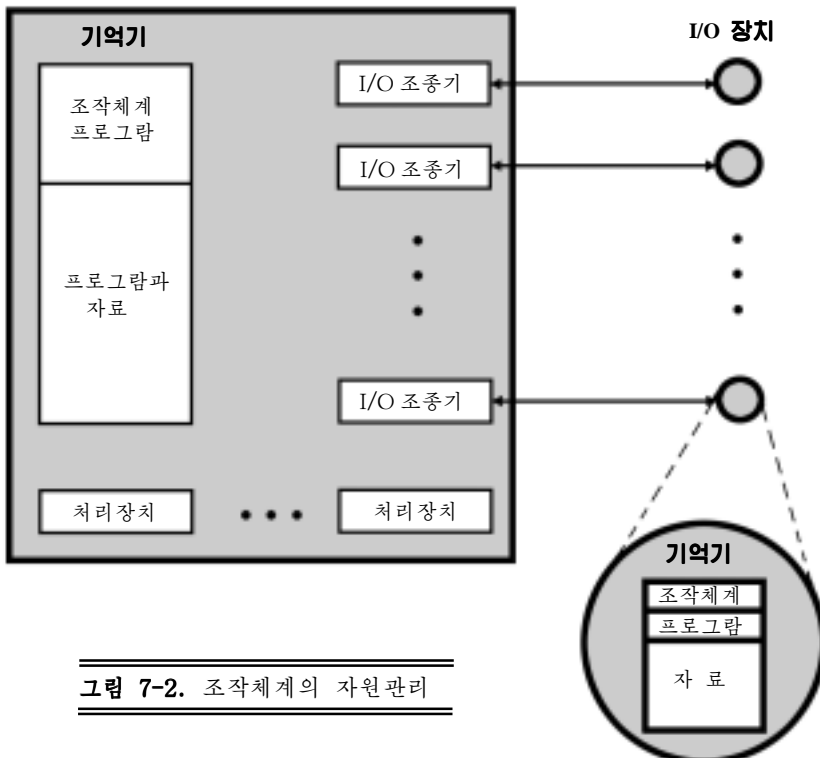



그림 7-2. 조작체계의 자원관리

주기억기에 있다. 이것을 **핵심부**라고 하는데 여기에는 조작체계에서 제일 많이 리용되는 기능들과 주어 진 시간에 현재 리용되고 있는 조작체계의 다른 부분들이 들어 있다. 이 자원(주기억기)의 할당은 앞으로 보게 되겠지만 조작체계와 처리장치에서의 기억기관리하드웨어에 의하여 공동으로 조종된다. 조작체계는 입출력장치가 언제 집행프로그람에 의해 리용되는가를 결정하며 파일의 사용과 파일에로의 접근을 조종한다. 처리장치 그자체는 자원이며 개별적인 사용자프로그람의 집행에 얼마만한 처리장치시간이 걸리는가는 조작체계에 의해 결정된다. 다중처리장치체계인 경우에 이러한 결정은 모든 처리장치들에서 진행해야 한다.

2. 조작체계의 형태

여러가지 류형의 조작체계를 구분하기 위하여 몇가지 주요특징들을 리용한다. 그 특징들은 두개의 독립적인 류형들로 나눌수 있다. 첫번째 류형은 체계가 묶음형인가, 호상작용형인가에 따르는 분류이다. **호상작용**체계에서 사용자 혹은 프로그람작성자는 일감의 집행을 요구하거나 처리요구를 집행하기 위하여 건반이나 영상표시장치와 같은 말단을 통하여 콤퓨터와 직접 대화를 진행한다. 더우기 사용자는 응용프로그람의 성질에 따라 일감이 집행되는 동안 콤퓨터와 통신을 진행할수 있다. **묶음처리**체계는 호상작용체계와 반대이다. 사용자프로그람은 다른 사용자로부터 제공되는 프로그람과 함께 묶음처리되며 콤퓨터조작수에 의하여 조작된다. 프로그람이 완료된 다음 결과는 사용자에게 인쇄되여 나온다. 오늘날 순수한 묶음처리체계는 거의 없다. 그러나 여기서 묶음처리를 간단히 설명하는것은 동시대의 조작체계를 리해하는데 어느 정도 필요할것이다.

두번째 류형은 체계가 **다중프로그람**방식을 리용하겠는가 안하겠는가에 따르는 분류이다. 다중프로그람방식은 처리장치가 한번에 여러개의 프로그람을 가능한껏 더 효률적으로 집행하게 하는것이다. 여러개의 프로그람들이 기억기에 넣어 지며 처리장치는 이것들을 빠른 속도로 절환하여 처리한다. 이와 다른 한가지는 단일프로그람방식체계로서 한번에 하나의 프로그람밖에 집행하지 않는다.

초기체계

1940년대 말부터 1950년대 중엽까지 기간에 나온 초시기 콤퓨터들에서는 프로그람작성자가 콤퓨터하드웨어와 직접대화를 진행하였는데 그 리유는 조작체계가 없었기때문이였다. 이 처리장치들은 영상표시장치, 빗장스위치, 여러가지 류형의 입구장치로 이루어 진 조종탁과 인쇄기로부터 조종을 받아 동작하였다. 처리장치명령으로 작성된 프로그람은 입력장치(즉 카드읽기장치)를 통하여 넣어 졌다. 만일 오유가 있어 프로그람이 정지되였다면 그 오유조건은 표시등을 켜는 방법으로 알려 주었다. 프로그람작성자는 등록기들과 주기억기들을 검사하여 오유원인을 결정하였다. 만일 프로그람이 정상적으로 완료되면 그 결과는 인쇄기에 출력되였다. 초기에 나온 이 체계들은 두가지 기본문제점을 제기하였다.

- **일정작성**: 모든 설치들은 처리장치시간을 예약하기 위하여 계약서를 사용하였다. 일반적으로 사용자는 30분의 배수로 된 시간블로크로 계약을 하였다. 사용자는 1시간 예약하고 45분내에 끝낼수 있다. 이것은 콤퓨터시간이 랑비되는것으로 되였다. 다른 한편 사용자는 할당된 시간내에 끝내지 못하는 문제들을 집행시킬수 있으며 이 경우에는 그 문제를 풀기전에 강제로 정지시켜야 하였다.
- **설치시간**: 일감이라고 하는 단일프로그람은 기억기에로 고급언어프로그람(원천프로그람)과 번역프로그람의 넣기, 번역된 프로그람(목적프로그람)의 기억, 목적

프로그람과 공동함수들을 함께 련결하는것을 포함하여야 한다. 이 매 단계들은 테프의 설치나 해제 혹은 카드설치를 포함할수 있다. 만일 오유가 발생하면 사용자는 일반적으로 설치순서의 시작으로 되돌아 가야 하였다. 따라서 생각하던 많은 시간이 집행을 위한 프로그람의 설치에 랑비되였다.

이 연산방식은 사용자가 순차로 콤퓨터를 접근한다고 하여 순차처리라고 불렀다. 이 시간이 지나감에 따라 여러가지 체계프로그람도구가 개발되면서 순차처리를 보다 효과적으로 진행하려는 시도들이 있었다. 이러한 도구들로는 모든 사용자들이 공동쏘프트웨어로 쓸수 있는 공동함수서고, 련결편집기, 적재기, 오유수정기, I/O 구동기루틴들이 있다.

간단한 묶음체계

초기의 처리장치들은 대단히 가격이 높았으므로 처리장치의 리용률을 최대로 하는것이 중요하였다. 일정작성과 설치시간때문에 랑비되는 시간은 무시할수 없었다.

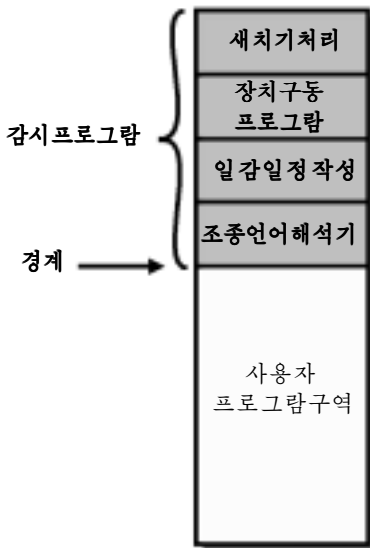



그림 7-3. 상주프로그람의 기억기배치

이 문제를 개선하기 위하여 단순한 묶음처리조작체계가 개발되였다. **감시프로그람**이라고도 하는 이러한 조작체계가 개발됨으로써 사용자는 더는 처리장치를 직접 접근하지 않아도 되게 되였다. 오히려 사용자는 감시프로그람을 리용하기 위하여 카드나 테프에 들어 있는 일감을 콤퓨터조작수에게 넘겨 주며 조작수는 그 일감들을 순서대로 묶어 그 전체 묶음을 입력장치에 넣었다.

이것이 어떻게 동작하는가를 리해하기 위하여 두가지 측면 즉 감시프로그람과 처리장치의 견지에서 고찰해 보기로 하자. 감시프로그람의 견지에서 고찰해 보면 사건의 처리과정을 조종하는것이 감시프로그람이다. 그러므로 감시프로그람의 대부분은 집행하는 동안 주기억기에 상주하고 리용될수 있어야 한다(그림 7-3). 이 부분을 **상주감시프로그람**이라고 한다. 감시프로그람의 나머지는 어떤 일감의 시작에서 사용자프로그람의 보조루틴으로 넣어 지는 편의프로그람이나 일반기능들로 이루어진다. 감시프로그람은 입력장치로부터(대표적으로 카드읽기장치 혹은 자기테프장치) 일감을 단번에 읽어 들인다. 읽어 들인 다음 현재일감이 사용자프로그람구역에 배치되고 조종이 이 일감에로 옮겨 진다. 일감이 완성되면 다음일감을 직접 읽어 들이는 감시프로그람에 조종을 넘긴다. 매 일감의 결과들은 사용자에게 넘겨 주기 위하여 출력된다.

처리장치의 견지에서 이 순서를 고찰하자. 어떤 시점에서 처리장치는 감시프로그람을 기억하고 있는 주기억기의 부분으로부터 명령을 집행한다. 이 명령들은 주기억기의 다른 부분에 다음일감을 읽어 들인다. 일단 한 일감이 읽어 지면 처리장치는 사용자프로그람시작에서 집행을 계속하도록 처리장치에 지시하는 분기명령을 감시프로그람에서 맞다들리게 된다. 처리장치는 프로그람의 집행끝이나 오유조건을 만날 때까지 사용자프로그람에서 명령을 집행한다. 어떤 사건이 감시프로그람에서 다음명령을 꺼낸다고 하자. 《조종이 일감으로 넘어 간다.》는 말은 처리장치가 현재사용자프로그람에서

명령을 꺼내여 집행한다는것을 의미하며 《조종이 감시프로그람으로 되돌아 간다.》는 말은 현재 처리장치가 감시프로그람으로부터 명령을 꺼내여 집행한다는것을 의미한다.

감시프로그람이 일정작성문제를 조종한다는것을 명백히 해야 한다. 일감묶음은 대기하게 되며 일감들은 빈 시간이 없이 가능한껏 빨리 집행된다.

일감이 시간을 어떻게 설정하는가? 감시프로그람은 이것을 원만히 수행한다. 매 일감에 대하여 명령들은 **일감조종언어**(JCL)에 포함된다. 이것은 감시프로그람에 명령들을 제공하기 위하여 리용되는 프로그람작성언어의 특수한 형태이다. 이 프로그람작성언어는 사용자가 FORTRAN 으로 작성된 프로그람과 그 프로그람에서 사용하는 자료에 의존하고 있는것과 같이 리용할수 있다. 매 FORTRAN 명령과 자료항목은 개별적인 카드나 테프의 개별적레코드에 기억된다. FORTRAN 과 자료행외에 일감은 "$"로 시작한 일감조종명령을 가지고 있다. 일감의 종합적인 양식은 다음과 같다.

```
$JOB
$FTN
•  ┐
•  ├ FORTRAN 명령
•  ┘
$LOAD
$RUN
•  ┐
•  ├ 자료
•  ┘
$END
```

이 일감을 집행하기 위하여 관리프로그람은 $FSN 선을 읽어 들이고 그의 기억기로부터 적당한 번역프로그람을 읽어 들인다. 번역프로그람은 사용자프로그람을 목적코드로 변환시키는데 그것은 기억기나 탄창기억기에 기억된다. 그것이 기억기에 기억되면 연산은《번역, 넣기, 집행》으로 된다. 테프에 기억되는 경우 $LOAD 명령이 필요하다. 번역동작후에 조종을 다시 얻는 감시프로그람이 이 명령을 읽어 들인다. 감시프로그람은 목적프로그람을 번역프로그람위치로 읽어 들이는 넣기프로그람을 기동시키며 그에 조종을 절환한다. 이 방법에서는 단 하나의 이런 보조체계가 남아서 집행된다고 하여도 주기억기의 한 토막이 여러개 보조체계들에서 공유될수 있다.

감시프로그람이나 묶음조작체계는 간단한 콤퓨터프로그람이라고 볼수 있다. 이것은 조종을 번갈아 주고받기 위하여 주기억기의 여러 부분들로부터 명령을 꺼내는 처리장치의 능력에 관계된다. 하드웨어의 다른 특성들에 대한 요구는 다음과 같다.

- **기억기보호:** 사용자프로그람이 집행중일 때 감시프로그람을 포함하는 기억기부분을 변경시키지 말아야 한다. 이런 시도가 생기면 처리장치하드웨어는 오유를 검출하고 감시프로그람에 조종을 절환한다. 그러면 감시프로그람은 일감을 중지하고 오유통보를 출구하며 다음일감을 읽어 들인다.
- **시계:** 시계는 독점체계로부터 단일일감을 막는데 리용된다. 시계는 매 일감의 시작에서 설정된다. 만일 시계가 멎었다면 새치기가 발생하며 조종은 감시프로그람에로 돌아 간다.
- **특권명령:** 어떤 명령들은 특권으로 구별되고 감시프로그람에 의하여서만 집행된다. 만일 처리장치가 사용자프로그람을 집행하는 동안 이와 같은 명령을 만난다

면 오유새치기가 발생한다. 특권명령들은 입출력명령이므로 감시프로그람은 모든 입출력장치들에 대한 조종을 맡아 진행한다. 실례로 이것은 사용자가 우연히 다음일감으로부터 일감조종명령을 읽는것을 막는다. 만일 사용자프로그람이 입출구수행을 요구하면 감시프로그람이 이 조작을 수행하게 하여야 한다. 사용자프로그람을 집행하는동안 특권명령을 만나면 처리장치하드웨어는 이것을 오유로 보고 감시프로그람에로 조종을 절환한다.

- **새치기:** 초기콤퓨터모델들은 이 능력을 가지지 못하였다. 이 특성은 조종을 중단하고 사용자프로그람으로부터 조종을 회복하는데 더 유연한 조작체계를 제공한다.

처리장치시간은 사용자프로그람의 집행과 감시프로그람의 집행사이에서 교체된다. 여기에는 두가지 상황이 존재한다. 즉 주기억기의 일부는 감시프로그람에 할당되며 일부 처리장치시간은 감시프로그람에 종사한다. 이것들은 둘다 우에서 고찰한 특성들로 이루어 진다. 이 특성들에 의하여 간단한 묶음체계는 콤퓨터의 리용률을 증대시킨다.

다중처리프로그람묶음체계

간단한 묶음조작체계가 제공하는 자동일감순서인 경우에 처리장치는 때때로 빈동작을 한다. 문제는 입출력장치가 처리장치에 비하여 뜨다는것이다. 그림 7-4는 표현식을 설명한다. 계산은 파일레코드를 처리하고 실현하는 프로그람에 집중된다. 이 실례에서 콤퓨터는 자료전환을 끝내는 입출력장치를 기다리는데 자기 시간의 96%를 소비한다. 그림 7-5 ㄱ는 이 상태의 실례이다. 처리장치는 입출력명령에 도달할 때까지 얼마만한 집행시간을 가진다. 그다음 계속 집행하기 앞서 입출력명령이 완료될 때까지 기다려야 한다.

이 비효과적인 과정은 필요 없다. 조작체계(상주하고 있는 감시프로그람)와 하나의 사용자프로그람을 기억시키는데는 충분한 기억기가 있어야 한다. 조작체계와 두개의 사용자프로그람을 기억시킬 공간이 있다고 가정하자. 한개 일감이 입출력요구를 기다릴 때 처리장치는 입출력을 기다리지 않는 다른 일감으로 전환할수 있다(그림 7-5 ㄴ). 더우기 3, 4 또는 그이상의 프로그람을 기억하고 그들사이에서 절환하도록 하기 위해서는 기억기를 확장해야 한다(그림 7-5 ㄷ).

하나의 레코드읽기	0.0015 초
100 개의 명령실행	0.0015 초
하나의 레코드쓰기	0.0015 초
총 시간	0.0031 초

$$\text{CPU 리용률} = \frac{0.0001}{0.0031} = 0.032 = 3.2\%$$

그림 7-4. 체계응용실례

그 처리를 **다중프로그람방식** 또는 **다중과제방식**이라고 한다. 이것은 현대조작체계의 기본주제이다.

다중프로그람체계의 우점을 설명하기 위하여 실례를 들어 보자. 256K의 기억기(조작체계가 쓰지 않는)와 디스크, 말단인쇄기를 가진 콤퓨터를 고찰하자. 3개의 프로그람 JOB1, JOB2, JOB3이 표 7-1에 표시한 속성을 가지고 같은 시간에 집행된다. JOB2,

JOB3 이 극소형처리장치를 요구하고 계속하여 JOB3 이 디스크와 인쇄기를 요구한다고 하자. 간단한 묶음환경에서 이 일감들은 련속적으로 집행된다.

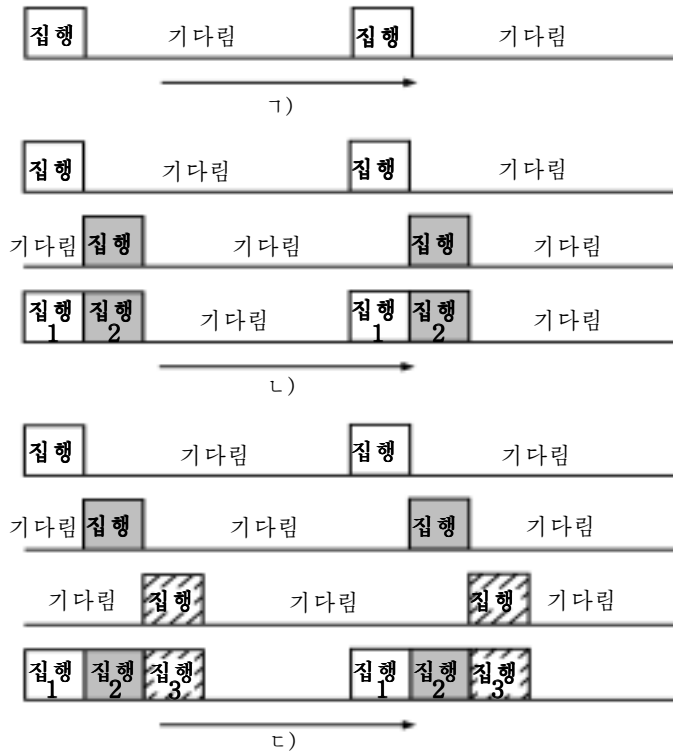



그림 7-5. 다중프로그람집행의 실례
ㄱ- 단일프로그람집행, ㄴ- 두개 프로그람의 다중집행,
ㄷ- 3 개 프로그람의 다중집행

따라서 JOB1 은 5 분동안에 완성되고 JOB2 는 5 분이 될 때까지 기다려야 하며 그때로부터 15 분후에 완성된다. JOB3 은 20 분후에 시작되며 그것이 시작된 때로부터 30 분동안에 완성된다. 평균자원리용, 생산액, 응답시간은 표 7-2 의 렬에서 보여 준다. 매 장치의 리용률은 그림 7-6 에서 보여 준다. 평균 30 분시간주기이상이 요구될 때 모든 자원들에 대하여 일반적으로 리용되지 않는다는것은 명백하다.

표 7-1. 간단한 프로그람집행속성

	JOB1	**JOB2**	**JOB3**
일감형태	복잡한 계산	입출력부하	입출력부하
수행시간	5 분	5 분	5 분
기억기요구	50K	100K	80K
디스크가 필요한가	필요 없음	필요 없음	필요 있음
말단이 필요한가	필요 없음	필요 있음	필요 없음
인쇄기가 필요한가	필요 없음	필요 없음	필요 있음

표 7-2. 원천리용에 대한 다중프로그람의 영향

	단일프로그람	다중프로그람
처리장치사용	17%	33%
기억기의 사용	30%	67%
디스크사용	33%	67%
인쇄기사용	33%	67%
경과시간	30 분	15 분
처리능률	6 개의 일감/한시간	12 개의 일감/한시간
응답시간	18 분	10 분

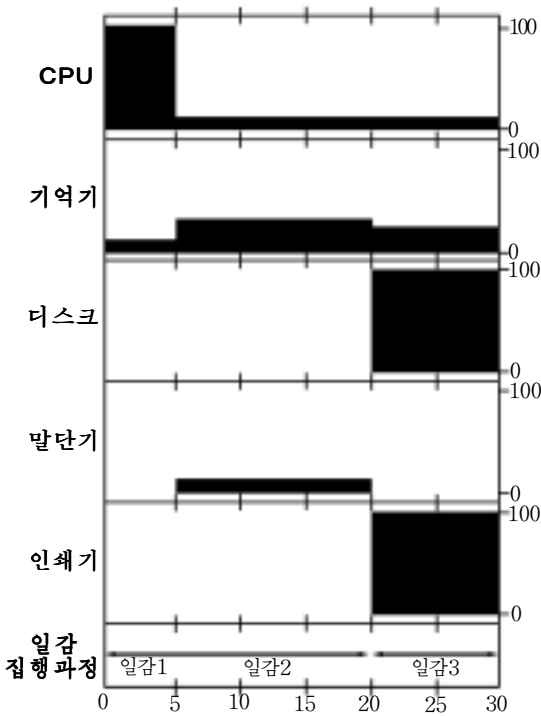



그림 7-6. 단일프로그람집행도표

다중프로그람방식의 조작체계에서 일감이 동시에 집행된다고 가정하자. 일감들사이에 약간의 자원경쟁이 있기때문에 3 개 일감들은 콤퓨터에 다른 일감들(JOB2 와 JOB3 이 그것들의 입출력조작작용을 위한 충분한 처리장치시간을 할당한다고 가정하는)이 존재하는 동안 거의 최소시간내에 집행할수 있다. JOB1 은 5분동안 완성할것을 요구하고 이것이 끝나면 JOB2 는 JOB1 의 1/3 시간,

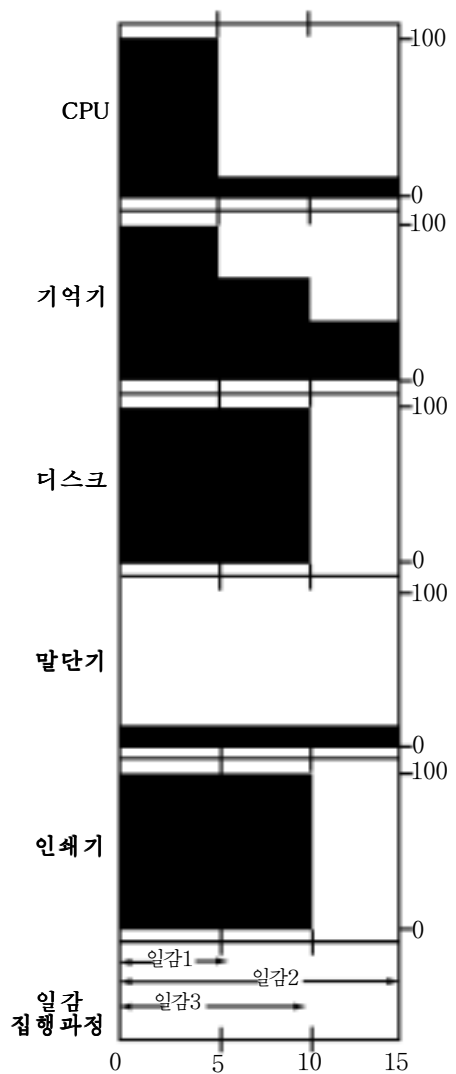



그림 7-7. 다중프로그람집행도표

JOB3 은 JOB1 의 1/2 시간동안에 끝난다. 세 일감은 15 분내에 끝나게 된다. 그림 7-7 에서 보여 준 히스토그람으로부터 얻은 표 7-2 의 렬은 다중프로그람을 시험할 때 얻은 것이다.

단순한 묶음체계와 비교해 보면 다중프로그람묶음처리체계는 일정한 콤퓨터하드웨어의 특성에 의존하지 않으면 안된다. 다중프로그람에 쓰이는 가장 주목할만한 부가적인 특징은 입출구새치기와 DMA 를 지원하는 하드웨어이다. 입출력을 구동하는 새치기와 혹은 DMA 를 가짐으로써 처리장치는 하나의 일감에 대한 입출력지령을 출력할수 있으며 입출력이 장치조종회로에 의하여 수행되는 동안 다음일감에로 집행을 옮길수 있다. 입출력조작이 완료될 때 처리장치는 새치기를 받으며 조종이 조작체계안의 새치기조종프로그람에로 넘어 간다. 조작체계는 이때 다른 일감에로 조종을 넘기게 된다.

다중프로그람조작체계는 단일프로그람 혹은 **단일프로그람작성**체계와 비교하면 명백히 복잡하다. 집행준비된 여러가지 일감들을 가지자면 일감은 주기억기에 있어야 하며 기억기관리의 일부 형식을 요구하여야 한다. 추가적으로 만일 여러 일감들이 집행준비되였다면 처리장치는 집행을 위한 한 프로그람을 결정해야 하는데 일정작성으로부터 일부 알고리듬이 요구된다. 이 개념들은 뒤에서 서술한다.

시분할체계

다중프로그람체계를 리용함으로써 묶음처리는 매우 효과적인것으로 되였다. 그러나 적지 않은 일감들은 사용자가 콤퓨터와 직접 대화하는 방식을 제공해 줄것을 요구한다. 실지로 업무처리와 같은 일부 작업들에 대한 대화방식(호상작용방식)은 본질적인 문제이다.

오늘날 대화식콤퓨터처리기구에 대한 요구는 전용마이크로콤퓨터를 리용하여 충족시키고 있다. 이러한 선택방식은 1960 년대에는 적용할수 없었는데 그 당시는 대부분의 콤퓨터들의 규모가 크고 값이 비쌌기때문이다. 그대신에 시분할체계가 개발리용되였다.

다중프로그람은 한번에 다중묶음일감들의 조종을 허락하며 호상작용하는 일감들을 다중으로 조종하는데 리용할수 있었다. 그후 이 수법을 시분할이라고 하였는데 그 리유는 처리장치시간이 다중사용자들에 의하여 공유되기때문이다. 시분할체계에서 다중사용자는 조작체계가 짧은 시간에 매 사용자프로그람의 집행 혹은 계산량을 처리하는것으로 말단을 통하여 동시에 체계를 접근할수 있다. 따라서 한번에 n 명의 사용자가 봉사를 요구한다면 매 사용자는 조작체계의 조종시간을 고려하지 않으며 유효한 콤퓨터속도를 1/n 로 볼수 있다. 그러나 상대적으로 사람의 반응시간이 느리므로 적합하게 설계된 체계에서 응답시간은 사용하는 콤퓨터에 알맞춤해야 한다.

묶음식다중프로그람방식과 시분할방식은 다중프로그람방식에서 리용한다. 기본차이점을 표 7-3 에 주었다.

표 7-3. 다중프로그람방식과 시분할방식의 차이점

	묶음다중프로그람	**시간공유**
원리적인 목적	최소처리장치사용	최소응답시간
조작체계에 의한 명령원천	일감과 제공된 일감조종언어명령	말단에 넣어 진 명령

제 2 절. 일정작성

다중프로그람작성에서 기본은 일정작성이다. 실제로 일정작성에는 4 가지 형태가 있다(표 7-4). 앞으로 이것들을 조사한다. 그러나 무엇보다 먼저 **프로쎄스**에 대한 개념을 소개한다. 이 용어는 1960 년에 Multics 조작체계설계에서 처음 쓰이였다. 이것은 **일감**보다 얼마간 더 일반적인 용어이다. 많은 정의들은 다음의것들을 포함하는 프로쎄스가라는 용어를 주고 있다.

- 집행상태의 프로그람
- 프로그람의“기동하는 상태”
- 처리장치가 할당된 실체

이 개념들은 고찰과정에 더 명백해 질것이다.

1. 긴시간일정작성

긴시간순서기는 체계에서 처리를 위하여 어느 프로그람이 승인되겠는가를 결정한다. 따라서 이것은 다중프로그람작성의 등급(기억기에서 많은 프로쎄스들)을 조종한다. 일단 승인되면 일감 혹은 사용자프로그람은 프로쎄스로 되고 짧은시간순서를 위한 대기렬이 증가된다. 일부 체계들에서 새롭게 만들어 진 프로쎄스는 중간시간순서를 위한 대기렬을 증가시키는 어떤 경우에 교환해 나가는 조건으로 된다.

체계묶음에서 혹은 일반조작체계부분의 묶음을 위해 새롭게 제출된 일감들은 디스크로 로정이 정해 지며 묶음대기렬에 유지한다. 긴시간일정작성은 그것이 가능할 때 대기렬로부터 프로쎄스들을 창조한다. 여기에 동반되는 두개의 결정이 있다. 우선 순서는 조작체계가 하나 혹은 그이상 추가적인 프로쎄스들을 가질수 있다는것을 결정하여야 한다. 두번째 순서는 접수하기 위한 일감 혹은 일감들을 결정하고 프로쎄스로 돌려야 한다. 리용되는 기준은 중요성, 기대되는 집행시간, 입출력요구를 포함한다.

시분할체계에서 호상 작용하는 프로그람들에 대하여 프로쎄스요구는 사용자가 체계와의 련결을 시도할 때 발생된다. 시분할사용자들은 단순히 대기렬을 늘이지 않으며 체계가 그것들을 접수할 때까지 기다림을 유지한다. 오히려 조작체계는 몇개의 미리 정의된 포화의 크기를 리용하여 체계가 포화될 때까지 권한을 부여한 사용자들을 모두 접수할수 있다. 이 시점에서 련결요구는 체계가 가득차 있으므로 사용자가 다시 시도해야 한다는 통보문과 맞다들게 된다.

표 7-4. 일정작성형태

긴시간일정작성	집행될수 있는 프로쎄스들의 묶음을 추가하는것을 결정
중간시간일정작성	주기억기에서 부분적으로 혹은 전부가 존재하는 프로쎄스의 [illegible]를 추가하는것을 결정
짧은시간일정작성	리용가능한 프로쎄스들이 처리장치들에 의하여 집행될수 있다는것[illegible] 결정
I/O 일정작성	프로쎄스의 미결상태에 있는 입출력요구가 리용가능한 입출력[illegible]치에 의하여 조종될수 있다는것을 결정

2. 중간시간일정작성

중간시간일정작성은 제3절에서 서술된 교환기능의 부분이다. 일반적으로 교환입구결정은 다중프로그람등급관리에 대한 요구에 기초한다. 가상기억을 사용하는 체계에서 기억기관리는 역시 흐름이다. 따라서 교환입구결정은 교환출구처리의 기억기요구를 고찰한다.

3. 짧은시간일정작성

높은준위일정작성은 상대적으로 드물게 집행되며 새로운 처리를 선택할것인가 말것인가 어느것을 취할것인가 하는 조잡한 결정들을 진행한다. 또한 **배포기**라고도 하는 짧은시간일정작성은 자주 집행되며 다음번에 어느 일감을 집행시키겠는가 하는 정확한 결심을 내리게 한다.

프로쎄스상태

짧은시간일정작성의 조작을 리해하기 위하여 프로쎄스상태의 개념을 고찰하여야 한다. 프로쎄스의 생존기간 그의 상태는 여러번 변한다. 어떤 시점에서 이 **상태**는 상태로 귀착된다. 주기상태는 그것이 그 시점에서 상태를 정의하는 어떤 정보가 존재한다는것을 암시하기때문에 리용된다. 최소로 프로쎄스에 대한 5개의 정의된 상태들이 있다(그림 7-8).

- **새로 시작:** 프로그람이 높은준위순서기에 의하여 허가되였지만 아직 집행을 위하여 준비되지 못함. 조작체계는 프로쎄스를준비상태로 옮겨서 초기화한다.
- **준비:** 프로쎄스는 집행을 위한 준비를 하여 프로쎄스에로의 접근을 기다린다.
- **집행:** 프로쎄스가 처리장치에 의해 집행되고 있다.
- **기다림:** 프로쎄스는 입출구와 같은 일부 체계에 대한 대기로 하여 집행을 중지한다.
- **중지:** 프로쎄스가 끌나게 되며 조작체계로부터 분리되여 무효로 된다.

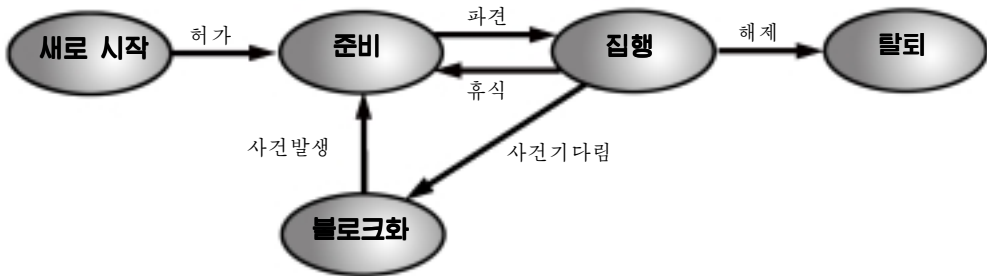



그림 7-8. 프로쎄스모델의 5가지 상태

체계에서 매 프로쎄스를 위하여 조작체계는 프로쎄스의 상태를 포함하고 있는 정보와 프로쎄스집행에 필요한 다른 정보들을 유지하여야 한다. 이 목적을 위하여 매 프로쎄스는 **프로쎄스조종블로크**(그림 7-9)에 의한 조작체계에서 나타나는데 일반적으로 다음의것들을 포함한다.

- **식별자:** 현재의 매개 프로쎄스가 유일한 지적자를 가진다.
- **상태:** 프로쎄스의 현재상태(new, ready 등)

- **우선권:** 관계가 있는 우선권준위이다.
- **프로그람계수기:** 집행되는 프로그람에서 다음명령의 주소가 있다.
- **기억기지적자:** 기억기에서 프로쎄스의 위치의 시작과 끝이다.
- **본문자료:** 프로쎄스가 집행되는 동안 프로쎄스에서 등록기들에 존재하는 자료가 있으며 그것들은 제 3 편에서 론의된다. 이 자료가 프로쎄스의 《문맥》을 표현한다고 말하는것은 가능하다. PC 에 추가되는 문맥자료는 프로쎄스가 준비상태를 벗어 날 때 복사된다. 그것들은 프로쎄스에 의하여 회복된다.
- **I/O 상태정보:**이 프로쎄스에 할당된 입출력요구들, 입출력장치(테프장치)와 처리장치에 할당된 파일일감 등이 포함된다.
- **계산정보:**처리장치시간, 박자시간, 극한시간, 계수값 등이 포함된다.

이 새 일감 혹은 집행을 위한 사용자요구를 시도할 때 그것은 빈 프로쎄스조종블로크를 창조하며 새로운 상태에서 프로쎄스와 련계를 맺는다.

그림 7-9. 프로쎄스조종블로크

일정작성기술

기억기에서 조작체계가 어떻게 여러가지 일감들에 대한 일정작성을 관리하는가를 알기 위해 그림 7-10 에서 간단한 실례를 생각해 보자. 이 그림은 주기억기가 어떻게 주어진 시점에서 구역분할되는가를 보여 준다. 물론 조작체계의 핵심부는 항상 상주한다. 부가적으로 A 와 B 를 포함하여 기억기부분에 할당된 많은 능동프로쎄스들이 있다.

프로쎄스 A 가 집행되는 시점에서 시작한다. 프로쎄스는 A 의 기억부분에 있는 프로그람으로부터 명령을 집행한다. 일부 그다음시점에서 프로쎄스는 A에서 명령을 집행하기 위해 중지하며 조작체계령역에서 명령을 집행한다. 이것은 3 가지 원인을 가진다.

- 프로쎄스 A 는 조작체계에로 봉사(말하자면 입출력요구)를 접근한다. A 의 집행은 이 접근이 조작체계에 의하여 만족될 때까지 중지된다.
- 프로쎄스 A 는 **새치기**를 발생한다. 새치기는 프로쎄스에로의 하드웨어발생신호이다. 이 신호가 검출될 때 프로쎄스는 A 집행을 중지하며 조작체계에 새치기조종을 전송한다. A 와 관련되는 여러가지 사건들이 중단을 일으킨다. 하나의 실례는 특권명령집행을 위한 시도와 같은 오유이다. 다른 실례는 요구시간(timeout)이다. 즉 처리장치점유로부터 어떤 하나의 프로쎄스를 막기 위해 매 프로쎄스는 짧은 기간에 처리장치를 허가한다.
- 주의할것은 요구하는 프로쎄스 A 와 관련이 없는 일부 사건들이 새치기를 발생시킨다는것이다. 실례로 입출력조작의 완료를 들수 있다.

어떤 경우에 결과는 다음과 같다. A프로쎄스조종블로크에서 처리장치는 현재의 상태자료와 A의 프로그람계수기를 복사한 다음 조작체계에서 집행을 시작한다. 조작체계는 입출구조작시작과 같은 시간일정작성부분이 프로쎄스다음에 집행할것을 결정한다. 이 실례에서 B를 선택하였다. 조작체계는 처리장치에 B의 상태자료를 기억하게 지시하며 B의 집행을 그만 두게 한다.

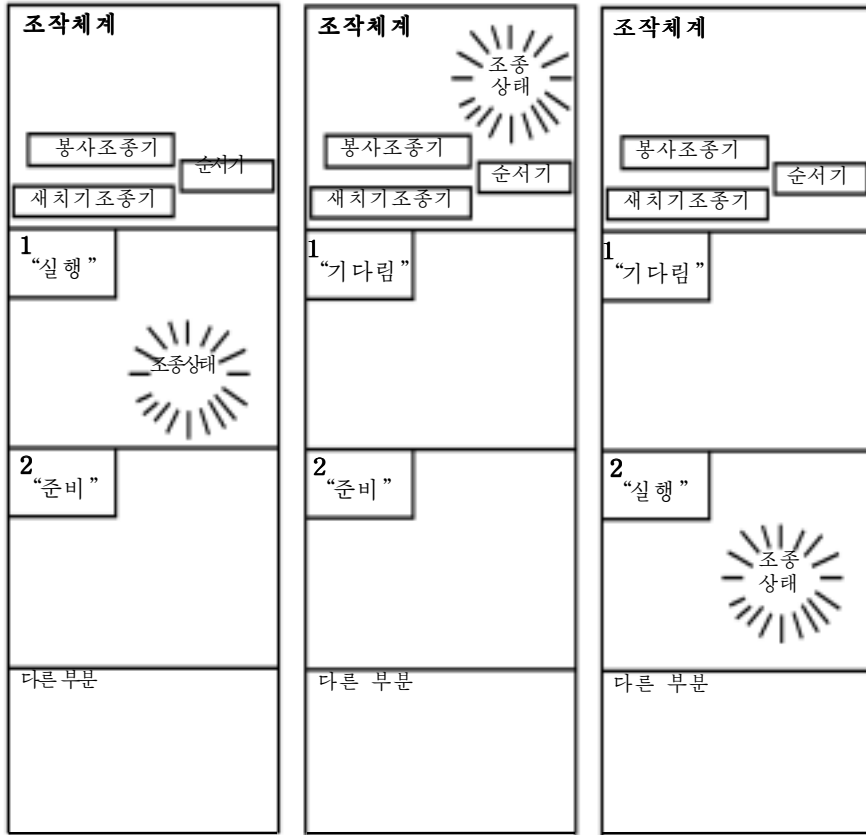



그림 7-10. 일정작성실례

이 간단한 실례는 짧은시간일정작성의 기본기능을 강조한다. 그림 7-11 은 다중프로그람과 프로쎄스일정작성을 포함하는 조작체계의 기본요소를 보여 준다. 조작체계는 새치기가 나타나면 새치기처리기에 처리장치의 조종을 주고 봉사접근이 나타나면 봉사접근처리기에 조종을 넘긴다. 일단 새치기나 봉사접근이 처리되면 짧은시간일정작성은 집행처리로 돌아 간다.

그 일감을 집행하기 위하여 조작체계는 대기렬의 수를 검사한다. 매 대기렬은 일정한 자원을 기다리는 처리들의 모임이다. 긴시간대기렬은 체계의 사용을 기다리는 모임이다. 그것은 짧은시간일정작성과 같이 형성된다. 일반적으로 매 처리에 일정한 시간을 주었다가 되돌리는 round-robin 알고리듬으로 수행된다. 우선순위도 사용된다. 끝으로 매 입출력장치에 대해서 입출구대기렬이 있다. 하나이상의 처리가 같은 입출구장치사용을 요구할수 있다. 매개 장치사용을 기다리는 모든 처리들은 그 장치의 대기렬에 놓인다.

그림 7-12 는 프로쎄스들이 조작체계의 조작밑에 콤퓨터에서 어떻게 진행되는가를 보여 준다. 매개 처리요구(묶음형일감, 사용자가 정의한 호상작용일감)는 긴시간대기

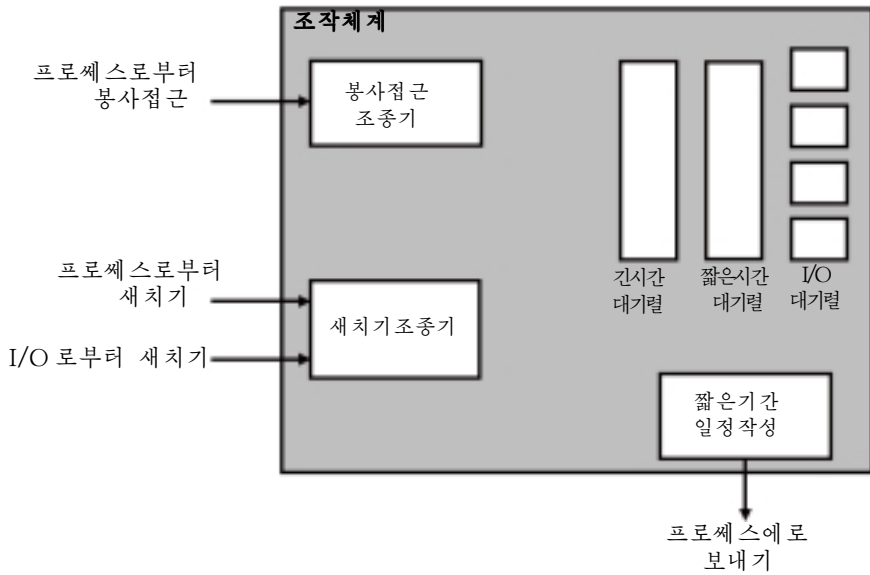



그림 7-11. 다중프로그람에서 조작체계의 기본구성요소

렬에 들어 간다. 자원이 사용가능하므로 처리요구가 처리로 되고 준비상태로 되며 짧은시간대기렬에 들어 간다. 처리장치는 집행하는 조작체계명령과 집행하는 사용자처리를 서로 번갈아 수행한다. 조작체계가 조종상태에 있을 때 짧은시간대기렬에 있는 어느 프로쎄스를 다음번에 집행시킬것인가를 결정한다. 조작체계가 자기의 과제를 끝내면 처리장치를 선택한 프로쎄스에로 넘긴다.

이미 언급된바와 같이 집행되는 프로쎄스는 여러가지 리유로 중지될수 있다. 프로쎄스입출구요구로 중지되면 해당한 입출구대기렬로 들어 간다. 요구시간이나 조작체계

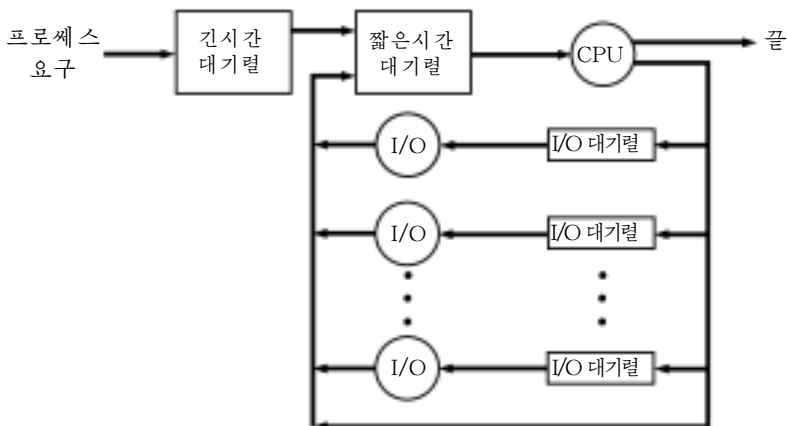



그림 7-12. 처리장치일정작성의 대기그림표현

가 긴급한 봉사에 응해야 하는것으로 인하여 프로쎄스가 중지되였다면 준비상태에 놓이며 짧은시간기다림렬에 넣어 진다.

끝으로 조작체계는 입출력대기렬도 관리한다. 입출력연산이 완성되면 조작체계는

입출구대기렬에서 해당 프로쩨스를 다음기다림프로쩨스로 선택하여 그 프로쩨스가 요구하는 입출력장치에 신호한다.

제 3 절. 기억기관리

단일사용자체계에서 주기억은 2개 부류로 갈라 지는데 하나는 조작체계를 위한것이고 다른것은 현재 집행되는 프로그람용이다. 다중사용자체계에서 기억기의 《사용자》부분은 여러 프로쩨스에 적용하여 편리하게 세분화된다. 이 세분화과제는 조작체계에 의해 동적으로 실현되며 이것을 **기억기관리**라고 한다.

효률적인 기억관리는 다중프로그람방식체계에서 가상기억방식이다. 적은 량의 프로쩨스만이 주기억기에 있다면 이때 모든 프로쩨스들의 대부분시간은 입출구대기로 되며 처리장치는 휴식상태에 있게 된다. 따라서 기억기는 가능한 여러개 처리를 기억기에 넣도록 효과적으로 배치하여야 한다.

1. 교환

그림 7-12에 대하여 다시 고찰하여 3가지 기다림형태들을 론의하자. 즉 새로운 프로쩨스에 대한 요구들을 등록하는 긴시간기다림렬, 처리장치의 리용을 준비하고 있는 프로쩨스들을 등록하는 짧은시간기다림렬, 처리장치의 리용에 준비되지 않은 프로쩨스들을 등록하는 여러가지 입출력기다림렬들이다. 이 정밀한 기계들은 입출력동작이 계산보다 더 느리므로 단일프로그람방식체계에서 처리장치는 많은 시간이 휴식상태로 된다.

그러나 그림 7-12 에서의 배렬은 이 문제를 완전히 해결하지 못한다. 이 경우에 기억기가 여러개 프로쩨스를 유지하고 있고 처리장치가 어떤 처리공정이 기다림상태에 있을 때 처리장치가 또 다른 프로쩨스로 이동할수 있다는것이 확실하다. 그러나 처리장치는 입출력보다 매우 빨라서 기억기에서의 모든 프로쩨스들은 입출력을 기다리는 상태에 있는것이 일반적인것으로 된다. 그리하여 다중프로그람방식에서 처리장치는 대부분 휴식상태에 있게 된다.

방도는 무엇인가? 주기억기는 확장될수 있으며 그리하여 더 많은 프로쩨스들을 가질수 있다. 그러나 이 방법에는 두가지 결함이 있다. 첫째로, 주기억기가 현재까지도 값이 비싼것이다. 둘째로, 기억기의 가격이 빨리 낮아 지는것만큼 기억기에 대한 프로그람들의 요구가 늘어 나는것이다. 따라서 큰 기억기는 많은 처리가 아니라 더 큰 처리를 초래하게 된다.

다른 해결은 그림 7-13 에서 서술한 **교환**이다. 일반적으로 디스크에 기억되는 프로쩨스요구들은 긴시간대기렬을 가진다. 이것들은 공간이 리용가능할 때 한번에 하나씩 주기억기에 들어 온다. 프로쩨스들이 수행되면 주기억기에서 없어 진다. 처리들이 완수되는데 따라 주기억밖으로 나간다. 그러면 기억기의 처리들이 모두 준비상태(모든것은 입출구연산에 대하여 기다림을 가진다.)에 있지 않는 상황이 생긴다. 빈상태를 유지하기 보다 처리장치는 이 처리들중 하나를 디스크에서 **중간대기렬**로 내보낸다. 이것은 일시적으로 기억기에서 나간 현존처리들의 대기렬이다. 조작체계는 중간대기렬에서 다른 처리를 가져 오거나 긴시간대기렬로부터 새로운 처리요구를 받아 들인다. 그러면 새로 도착한 처리가 집행이 계속된다.

그러나 교환은 입출력조작이며 따라서 문제를 더 좋게가 아니라 나쁘게 할 가능성이 있다. 그러나 디스크입출력은 일반적으로 체계입출력보다 빠르므로 교환이 보통 동작을 제고한다. 가상기억기를 포함하여 더 기발한 착상이 간단한 교환동작을 개선한다. 이것은 인차 론의된다. 그러나 먼저 구획분할과 페지화에 대한 설명이 있어야 한다.

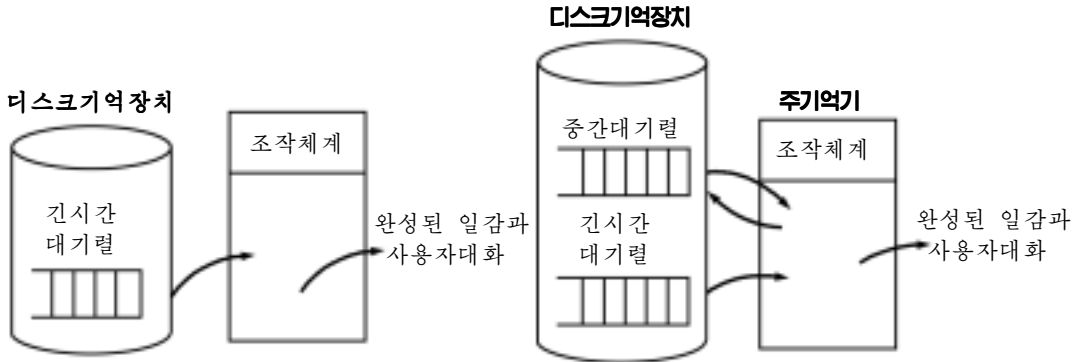



그림 7-13. 교체하기의 리용
ㄱ-간단한 일감일정작성, ㄴ-교체하기

2. 구획분할

사용가능한 기억기의 가장 간단한 구획분할방법은 그림 7-14와 같은 고정크기 구획을 사용하는것이다. 구획들이 고정된 크기이지만 그것들이 같은 크기를 요구하는것은 아니다. 한개 처리가 기억기로 들어 오면 그것은 기억시킬 가능한 가장 작은 구획을 차지한다. 크기가 다른 고정크기구획을 사용하여도 기억기쪼각이 있다. 많은 경우 처리는 구획이 제공한 만큼 큰 기억기를 요구하는것은 아니다. 실례로 3Mbyte 기억기를 요구하는 처리는 4M 구획을 차지하는데(그림 7-14 ㄴ) 1M 쪼각은 다른 처리가 사용한다.

더 효과적인 방법은 **가변크기구획**을 사용하는것이다. 처리가 기억기에 들어오면 그것은 정확히 그가 요구하는것만한 기억기를 요구하며 그이상은 필요 없다. 그림 7-15 에 1MB 주기억을 사용하는 실례를 보여 주었다. 처음에 주기억기는 조작체계구역을 제외하고는 비여 있다(ㄱ). 첫 세개의 프로쎄스들의 조작체계가 끝나는 위치로부터 시작하여 매개 처리가 충분한 공간을 차지한다(ㄴ,ㄷ,ㄹ). 이때 4

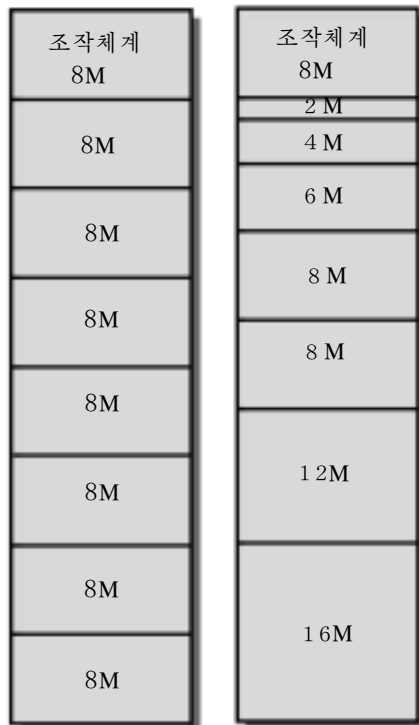



그림 7−14. 64MB기억기의고정분배실례
ㄱ)−같은크기의 분배, ㄴ)−서로다른크기의 분배

번째 프로쎄스가 들어 앉기에는 지내 작은 《공간》이 기억기끝에 존재하게 된다. 어떤 점에서도 기억기의 아무 처리가 준비되지 않는다. 조작체계는 새로운 프로쎄스인 프로쎄스 4 를 넣기 위해 충분한 공간을 가지는(ㅂ) 프로쎄스 2 를 교체하여 내보낸다(ㅁ). 프로쎄스 4가 프로쎄스 2보다 작으므로 다른 작은 공간이 생긴다. 다음 주기억기에 있는 그 어떤 프로쎄스도 준비되여 있지 않고 프로쎄스 2 가 준비대기상태에 놓이게 되는 시점에 도달할수 있다. 프로쎄스 2 를 위한 충분한 기억기가 없으므로 조작체계는 프로쎄스 1 을 내보내고(ㅅ) 프로쎄스 2 를 받아 들인다(ㅇ). 이 실례에서 보는바와 같이 이 방법은 시작은 잘되였지만 결국 기억기에 많은 작은 공간을 가지게 된다. 시간이 흐름에 따라 기억기는 더욱 분할화되고 기억기리용률이 떨어 진다. 이 문제를 해결하기 위한 한가지 기술이 **모으기**이다. 조작체계는 모든 기억기자유공간이 하나의 블로크로 되도록 주기억안에서 처리들을 밀어 놓는다. 이것은 시간을 소비하는 공정이며 처리장치 시간을 많이 소비한다.

구획분할의 결함을 취급하기전에 하나의 미해결문제를 명백히 해야 한다. 만일 독자가 잠시 그림 7-15 를 고찰해 보면 프로쎄스가 교체되여 들어 올 때마다 주기억기의 같은 위치에 넣어지지 않는다는것이 명백해 진다. 그럼에도 불구하고 모으기를 사용하면 처리는 주기억기에서 밀리운다.

기억기에 있는 처리는 명령과 자료로 이루어 진다. 명령들은 두가지 형태의 기억기

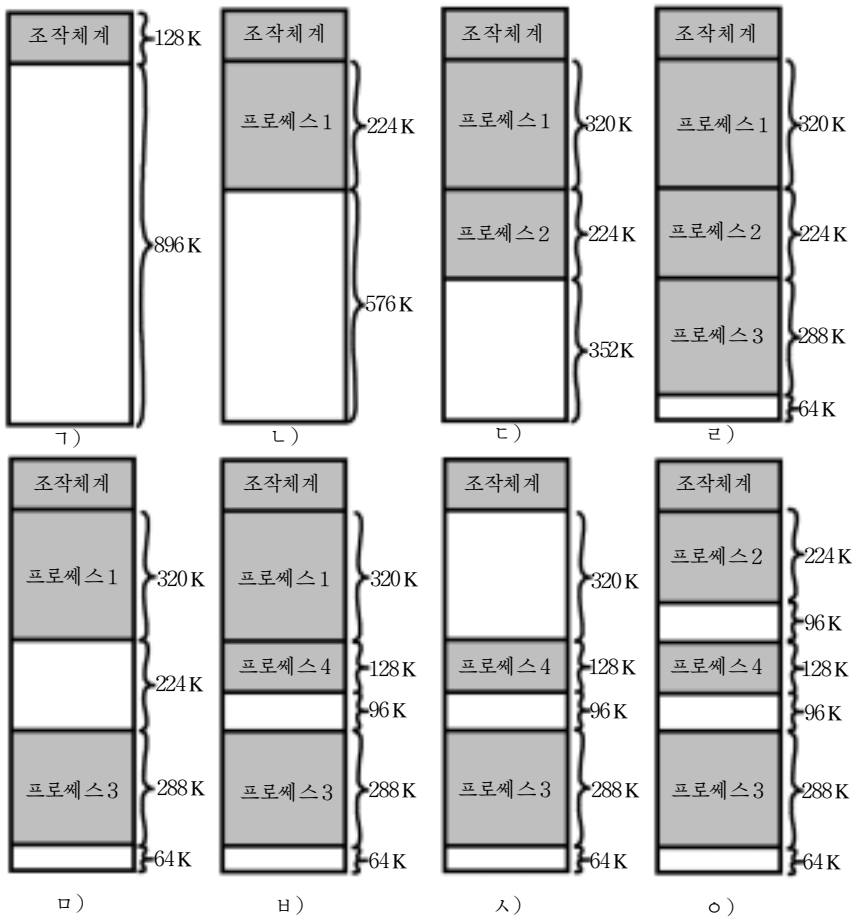



그림 7-15. 동적배치의 영향

위치주소를 포함하고 있다.

- 자료항목의 주소
- 분기명령에서 리용되는 명령의 주소

그러나 이 주소들은 고정이 아니다. 그것들은 처리가 교환되여 들어 올 때마다 변한다. 이 문제를 풀기 위하여 론리주소와 물리주소를 구분한다. **론리주**소는 프로그람의 시작으로부터의 상대위치로 표현된다. 프로그람에서의 명령들은 론리주소만을 가진다. **물리주소**는 주기억에서의 실지주소이다. 처리장치가 어떤 처리를 집행할 때 그것은 기준주소라고 하는 처리의 시작위치에 매개 론리주소를 더하여 론리주소를 물리주소로 자동적으로 변환한다. 이것이 처리장치의 장치적특징의 다른 한가지 실례이다. 이 장치적특징의 정확한 성질은 기억기관리에 의존한다.

3. 페지화

갈지 않은 고정크기 및 가변크기구획은 다 기억기사용에 비효과적이다. 그러나 기억기를 상대적으로 작은 같은 고정된 크기의 덩어리로 가르고 매개 처리도 일정한 크기의 작은 고정크기덩어리로 나눈다고 가정한다. 그러면 프로그람의 덩어리는 **페지**라고 하는데 **프레임** 또는 페지프레임이라고 하는 기억기의 작은 덩어리에 할당될수 있다. 그러면 그 처리에서의 기억기쪼각은 마지막페지에만 생긴다.

그림 7-16 은 페지와 프레임의 실례를 보여 준다. 어떤 주어 진 시점에서 기억기의 일부 프레임들은 쓰이고 일부는 자유상태에 있게 된다. 자유상태프레임들에 대한 표가 조작체계에 의해 유지된다. 디스크에 기억된 처리 A 는 4 개 페지로 이루어 져 있다. 이 처리를 넣기할 때가 되면 조작체계는 4 개의 빈 프레임을 찾아서 처리 A 의 4 개 페지를 4 개 프레임에 적재한다. 이 실례에서처럼 처리를 받아 들일 빈프레임이 불충분하다고 가정하자. 조작체계가 처리 A 의 적재를 그만둘것인가? 그것은 아니다. 왜냐하면 론리주소라는 개념을 다시 쓸수 있기때문이다. 간단한 기준주소는 전혀 만족하지 않는다. 조작체계는 매개 처리들에 대해 페지표를 검사한다. 페지표는 처리들의 매개 페지에 대한 프레임위치를 보여 준다. 프로그람안에서 매개 론리주소는 페지번호와 그 페지안에서의 상대주소로 이루어 져 있다. 간단한 구획분할인 경우 론리주소는 프로그람의 시작으로부터의 상대위치인데 처리장치는 그것을 물리주소로 변환한다. 페지화에서 론리-물리주소변환은 여전히 처리장치가 장치적으로 수행한다. 처리장치는 현재처리의 페지표를 어떻게 접

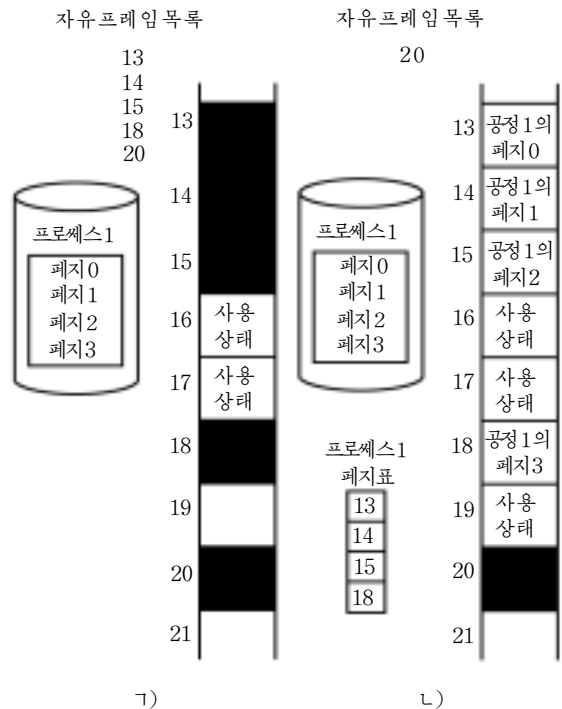



그림 7－16. 자유프레임들의 배치: ㄱ-전, ㄴ-후

근해야 하는가를 알고 있어야 한다. 론리주소(페지번호, 상대주소)를 가지고 처리장치는 물리주소(프레임번호, 상대주소)를 내기 위해 페지표를 사용한다. 그 실례는 그림 7-17 이다.

그 문제를 해결하는 방법은 이미 나왔다. 주기억기는 많은 같은 크기의 작은 프레임로 나누어 진다. 매개 프로쎄스들은 프레임크기의 페지들로 나누어 진다. 작은 처리는 적은 페지를 요구하고 큰 처리는 많이 요구한다. 어떤 처리가 들어 오면 그의 페지들은 가능한 프레임들에 넣기되고 페지표가 설정된다.

4. 가상기억기

요구페지화

페지화를 사용하면 다중사용자체계에 실지로 효과적이다. 더우기 프로쎄스를 페지로 갈라 놓는 간단한 전략으로부터 또 다른 중요한 개념 즉 가상기억기가 발전하게 되였다.

가상기억기를 리해하려면 이미 론의한 페지화개념을 재정의해야 한다. 그 재정의는 **요구페지화**이다. 어떤 처리의 매개 페지는 그것이 필요하여야 기억기에 들어 온다는것이다(즉 요구).

긴 프로그람과 많은 수의 자료로 이루어 지는 큰 처리를 고찰하자. 짧은시간주기동안에 집행은 프로그람의 작은 부분(보조루틴)에서 진행되며 자료가 한두개만 사용될것이다. 이것이 부록 4 에서 소개한 국부성의 원리이다. 프로그람이 일시 중지되기전에 몇개 페지만이 리용되고 있는 프로쎄스에 대하여 그의 모든 페지들을 주기억기에 넣는 것은 명백히 랑비로 된다. 우리는 몇개의 페지만을 넣기함으로써 기억기를 더 효과적으로 리용할수 있다. 만일 프로그람이 주기억기에 없는 페지의 명령으로 분기되거나 프로그람이 기억기에 없는 페지의 자료를 참조한다면 **페지오유**가 일어 난다. 그래서 조작체계가 필요한 페지를 들여 오게 한다.

따라서 임의의 시점에서 주어 진 처리의 일부 페지가 기억기에 존재하므로 더 많은 프로쎄스들이 기억기에 있을수 있다. 그러나 리용하지 않은 페지를 내보내고 들여 오는데 시간이 걸린다. 조작체계는 이 방법을 어떻게 관리하는가를 잘 알아야 한다. 조작체계가 한개 페지를 들여 올 때 다른 페지를 내보내야 한다. 만일 직전에 사용된 페지를 내보내면 그 페지를 인차 다시 써야 할 때가 있을것이다. 이런것은 많은 경우 과도교체라고 하는 상태를 초래한다. 처리장치는 명령집행보다 페지교환에 더 많은 시간을 쓸것이다. 이 과도교체의 해결을 1970 년대에 많이 연구했으며 여러가지 복잡하면서도 효과적인 알고리듬을 내놓았다. 명백히 조작체계는 최근경력에 기초하여 최근에 사용한 페지를 놔두게 한다.

요구페지화로 하여 전체 페지를 주기억기에 넣어야 할 필요가 없다. 이 사실은 현저한 차이를 가져 왔다. **프로쎄스가 주기억기보다 더 클 가능성이 있다**. 프로그람작성에서 가장 기본적인 결함의 하나가 제기된다. 요구페지법이 없이 프로그람작성자는 어느만한 기억기가 사용가능한가를 정확히 알아야 한다. 요구페지화로 하여 그런 일감이 조작체계와 장치에 넘겨 진다. 프로그람작성자와는 관계없이 사람들은 디스크기억기와 같은 크기로 기억기를 끝없이 취급한다.

처리가 주기억기에서만 집행되므로 그 기억기는 **실지기억기**로 취급된다. 그러나 프로그람작성자와 사용자는 다르게 할당된 매우 큰 기억기를 느낀다. 후자는 따라서 가상기억기로 취급된다. 가상기억기는 다중프로그람작성자에게 매우 효과적이며 주기억기의 제한을 느끼지 않게 한다.

페지표구조

기억기에서 단어를 읽어 내는 기본적인 기구는 페지번호와 편위로 이루어 진 가상주소 혹은 론리주소를 페지표를 리용하여 얻은 프레임번호와 편위로 이루어 진 물리주소로 변환하는 기능을 가진다. 페지표가 처리의 크기와 관련되는 가변길이이므로 그것을 등록기에 들수 없다. 대신 그것은 접근할수 있는 주기억기에 있어야 한다. 그림 7-17 에서는 이 방식의 장치적실현을 보여 주었다. 어떤 특별한 처리가 집행될 때 어떤 등록기가 그 처리의 페지표시작주소를 가리킨다. 가상주소의 페지번호를 그 페지의 첨수로 사용하고 대응한 프레임번호로 취급한다. 이것은 필요한 실주소를 만들기 위하여 가상주소의 편위부분과 결합된다.

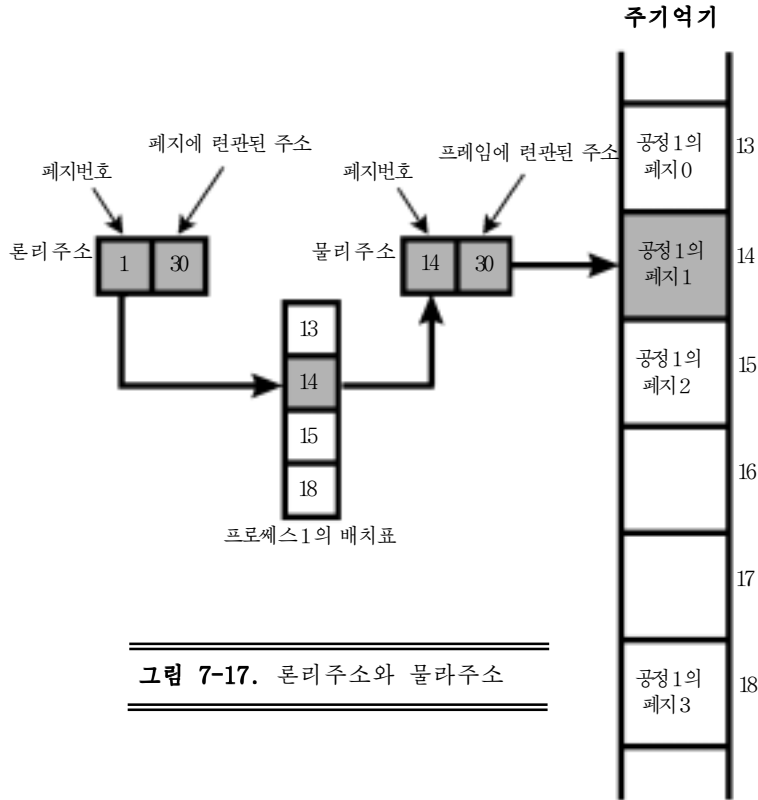



그림 7-17. 론리주소와 물라주소

대부분 체계에는 처리당 한개 페지표가 있다. 그러나 매 처리는 커다란 크기의 가상기억을 처리한다. 실례로 VAX 구성방식에서 매개 처리는 2^{31}=2GB 의 가상기억기를 가질수 있다. 2^9=512byte 페지를 사용하면 매 **처리당** 222 개의 페지표항목이 요구된다. 명백히 페지표에 쓰인 기억기는 무시할수 없을 만큼 크다. 이 문제를 해결하기 위해 모든 가상기억토막들은 페지표를 실기억기가 아니라 가상기억기에 보관한다. 이것은 페지표가 다른 페지들처럼 페지화의 대상이라는것을 의미한다. 어떤 처리가 집행중에 있을 때에는 적어도 그 페지표의 한 부분이 주기억기에 있어야 한다. 주기억기에는 현재집행페지의 페지표기입이 포함된다. 일부 처리장치들은 큰 페지표를 조직하기 위해 2 준위방식을 사용한다. 이 방식에는 매개 기입이 하나의 페지표를 가리키는 페지등록부가 있다. 따라서 등록부의 길이가 X 이고 페지표의 최대길이가 Y 이면 처리는 X×Y 개의 페지로 이루어 질수 있다. 일반적으로 페지표의 최대길이는 한페지로 제한된다.

이 장의 뒤에서 Pentium II를 고찰할 때 이 2준위방법을 고찰한다.

1 혹은 2 준위페지표를 리용하는 다른 한가지 방법은 반전페지표구조를 리용하는것이다(그림 7-18). 이 방법은 IBM의 AS/400과 PowerPC를 포함한 모든 IBM의 RISC 제품들에서 사용된다.

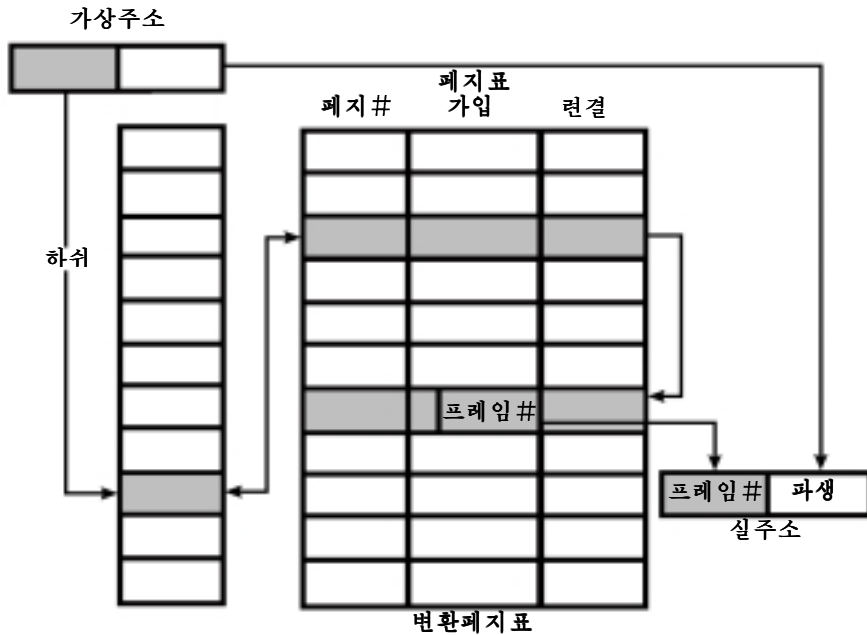



그림 7-18. 변환페지표의 구조

이 방법에서 가상주소의 페지번호부분은 간단한 하쉬함수[1]을 리용하여 하쉬표에 배치된다. 하쉬표는 페지표항목을 포함한 반전페지표에 대한 지시기를 가진다. 이 구조에서는 가상페지당 하나의 항목을 포함하기보다는 하쉬표와 매개 실기억기페지에 대한 반전페지표에 하나의 항목이 존재한다. 따라서 실지기억기의 고정된 부분이 처리나 가상페지들의 수를 고려함이 없이 표에 요구된다. 하나이상의 가상주소가 같은 표항목에 배치되므로 자리넘침을 관리하는데 련쇄기술을 리용한다. 이 하쉬기술은 전형적으로 하나 아니면 2개 항목으로 된 짧은 련쇄로 실현한다.

5. 변환엿보기완충기

원리적으로 모든 가상기억참조는 두번의 물리적기억기접근을 하게 하는데 하나는 근사한 페지표항목을 꺼내고 다른 하나는 필요한 자료를 꺼내는것이다. 따라서 정확한 가상기억방식은 2배의 기억접근시간을 가진다. 이 문제를 해결하기 위해 많은 가상기억방식들은 TLB라고 하는 페지표항목들을 위한 특수고속완충기를 사용한다. 이 완충

[1] 하쉬함수는 0부터 M까지의 수들을 0부터 N의 수에로 넘긴다. 여기서 M>N이다. 하쉬함수의 출력은 하쉬표에 색인으로서 리용된다. 하나이상의 입력이 같은 출력에 넘겨짐으로 하나의 출력항목에 관하여 하쉬표입구점 즉 이미 차지되여 있는 하쉬표입구점에로 넘길수 있다. 그 경우에 새로운 항목은 다른 하쉬표배치에로 넘어 가야 한다. 일반적으로 새로운 항목은 첫 련속적인 빈공간에 위치하며 본래의 배치의 지적자는 입구점들을 서로 련결시켜 준다(하쉬표에 대한 보다 자세한 론의를 위해서는 [STAL98]을 볼것).

기는 기억기 캐쉬와 같은 기능을 수행하며 최근에 사용된 페지표항목들을 기억한다. 그림 7-19 는 TLB 사용의 흐름도를 보여 준다. 국부화의 원리에 의해 많은 가상기억참조는 최근 사용된 페지의 위치로 된다. 그러므로 모든 참조는 그 고속완충기에서 페지표항목들을 포함하게 된다. VAX TLB 에 대한 연구는 이 방식이 충분히 개선된 성능을 가진다는것을 보여 준다[CLAR85, SATY81].

가상기억기구는 이 고속완충기체계와 호상작용을 해야 한다. 이것은 그림 7-20 에 보여 주었다. 가상주소는 일반적으로 페지번호와 편위의 형태로 이루어 진다. 먼저 기

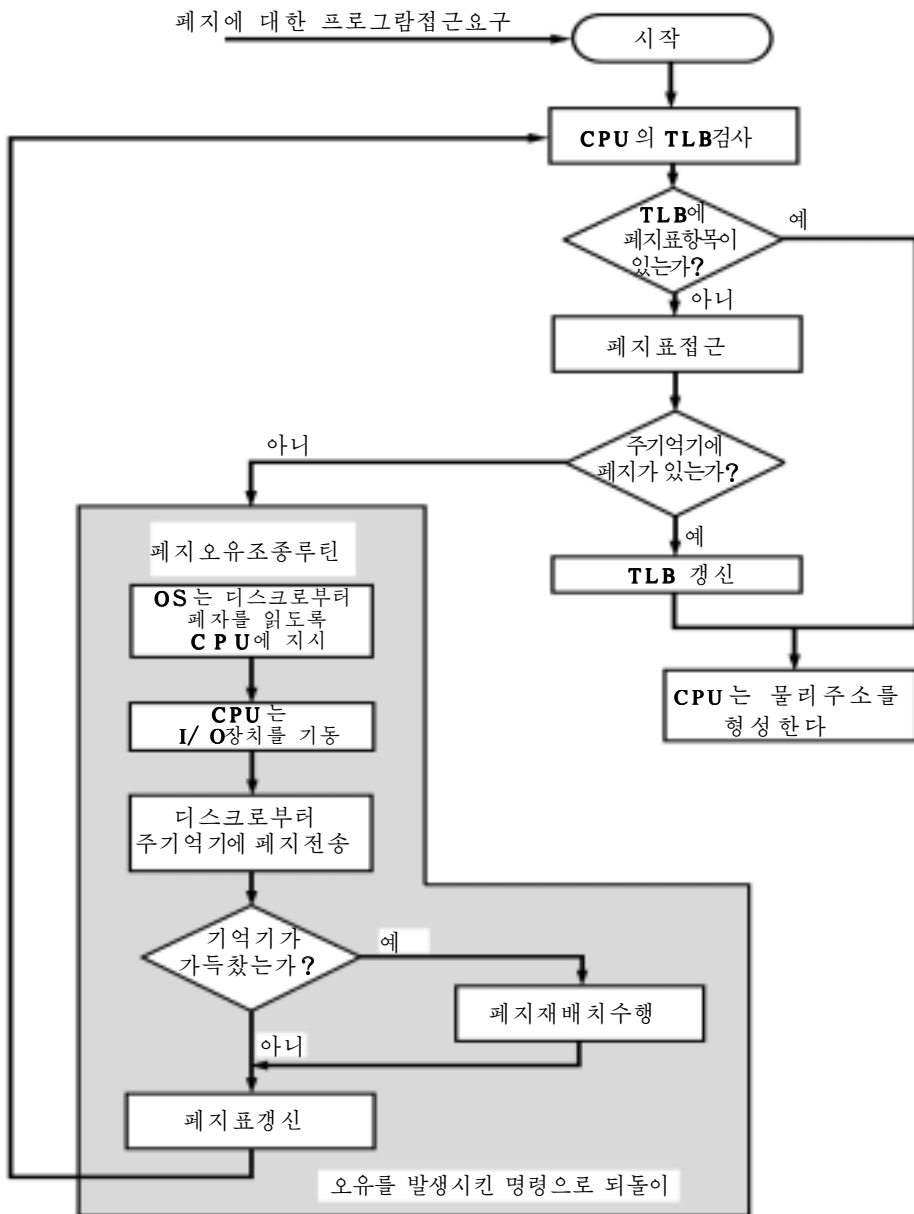



그림 7-19. 페지화와 변환완충기의 조작

억기체계는 같은 페지표항목이 있는가를 보기 위하여 TLB 를 참조한다. 만일 있다면 실지(물리)주소는 프레임번호와 편위를 결합하여 얻어 진다. 만일 없으면 항목이 페지표로부터 접근된다. 일단 표식자와 나머지부분으로 형성되는 실주소가 얻어 지면(그림 4-17 참고) 캐쉬는 단어가 존재하는 블로크가 존재하는가를 찾아 본다. 만일 있다면 처리장치로 되돌아 간다. 그렇지 않다면 주기억기로부터 단어를 캐쉬에 받아 들인다.

독자들은 단일기억기참조를 진행하는 처리장치하드웨어의 복잡성을 충분히 평가할수 있다. 가상주소는 실주소로 변환된다. 이것은 TLB 나 주기억기, 디스크에 있는 페지표에 대한 참조를 진행한다. 후자의 경우에 단어를 포함하는 페지는 주기억기에 넣어 져야 하며 이 페지의 블로크는 캐쉬에 넣어 진다. 추가적으로 이 페지에 대한 페지표항목은 갱신되여야 한다.

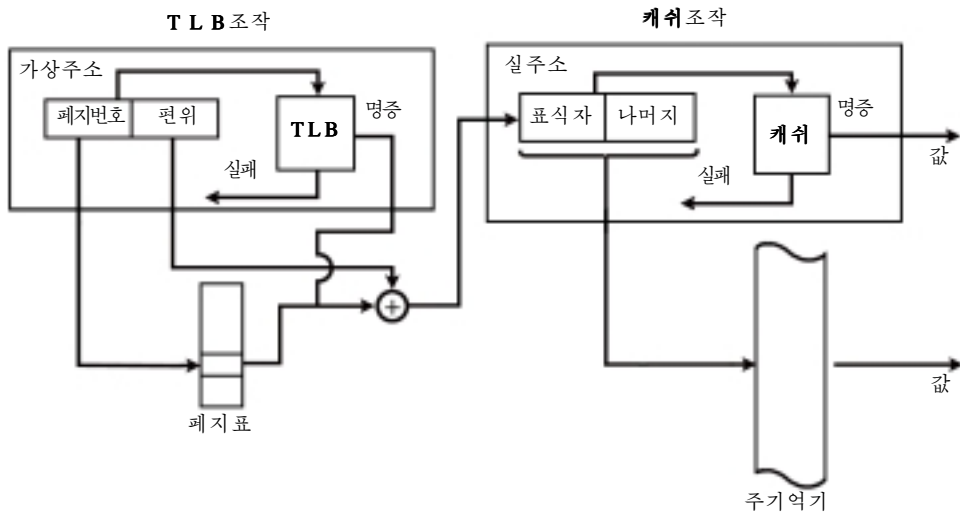



그림 7-20. 변환엿보기완충기와 캐쉬조작

6. 토막화

주소가능한 기억기를 세분화할수 있는 또 다른 방법에는 토막화가 있다. 페지화는 프로그람작성자에게 보이지 않으며 프로그람작성자에게 더 큰 주소공간을 제공할 목적으로 봉사한다면 토막화는 프로그람작성자에게 보이며 프로그람과 자료구성의 편리성, 명령과 자료에 대한 특권과 보호속성들을 결합시키는 수단들을 제공한다.

토막화는 프로그람작성자에게 기억기가 여러개의 주소공간이나 토막으로 이루어 진것처럼 보이게 한다. 토막은 가변이며 실지로 동적크기이다. 일반적으로 프로그람작성자나 조작체계는 프로그람과 자료를 서로 다른 토막들에 할당한다. 여러가지 형태의 프로그람에 많은 프로그람토막과 자료토막이 있을수 있다. 매개 토막은 접근권한을 부여 받는다. 기억기참조는 주소형식(토막번호, 편위)으로 이루어 진다.

이 구성은 프로그람작성자들에게 비토막화주소공간에 비하여 많은 우점을 제공해준다.

- 자료구조의 확장에 대한 조종이 간단하다. 프로그람작성자가 개별적인 자료구조가 얼마나 크게 형성될것인가를 미리 알지 못한다면 그것을 추측할 필요가 없다.

그것은 늘어 나는 자료구조취급을 간단하게 한다. 자료구조는 자기의 토막에 할당될수 있으며 조작체계는 토막을 필요에 따라 확장, 축소시킨다.
- 이 구성은 전체 프로그람묶음이 재련결되고 재넣기되는것을 요구함이 없이 프로그람들이 독자적으로 변경되고 재콤파일되게 한다. 또한 여러가지 토막들의 리용을 이룩할수 있다.
- 이 구성자체가 프로쎄스들속에서 공유되게 한다. 프로그람작성자는 편의프로그람이나 다른 프로쎄스들에 의하여 주소화될수 있는 토막내에 유효한 자료표를 배치할수 있다.
- 보호에 적합하다. 토막이 프로그람들이나 자료의 잘 정의된 묶음을 포함하도록 구축될수 있으므로 프로그람작성자나 체계관리자가 편리한 형태로 접근특권을 할당할수 있다.

이 우점들은 프로그람작성자에 의해서 볼수 없는 페지화에서는 리용할수 없다. 다른 한편으로 페지는 기억기관리의 효률적인 형태를 제공한다는것을 알수 있다. 두 우점들을 결합하기 위하여 일부 체계들은 페지화나 토막화를 제공하는 하드웨어와 조작체계쏘프트웨어를 장비한다.

제 4 절. Pentium Ⅱ와 PowerPC의 기억기관리

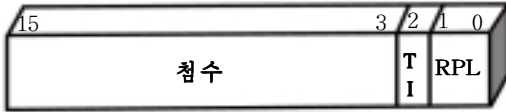
1. Pentium Ⅱ의 기억기관리장치

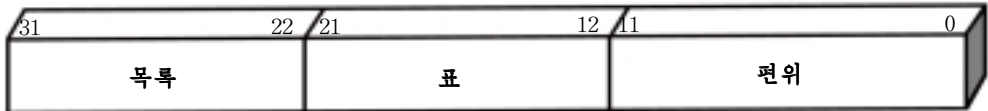
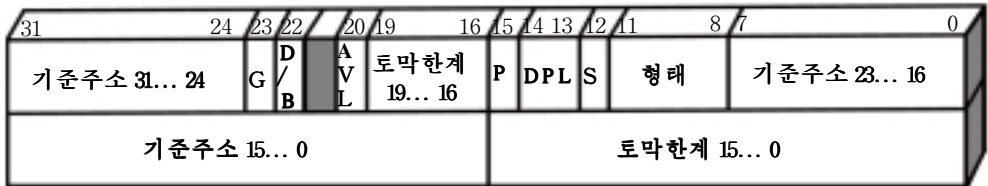
32bit 구성방식으로 소개된 극소형처리장치들은 중규모, 대규모체계들에서 배운 절들에서 취급된 기억기관리방법들을 전개하고 있다. 많은 경우 극소형처리장치는 이전의 대형체계에 비해 우월하다. 그 방식들은 극소형처리장치하드웨어판매자들에 의하여 발전되며 여러가지 조작체계들에 제공되기때문에 아주 일반적인 추세로 되고 있다. PENTIUM Ⅱ에서 사용되는 방식이 대표적인 실례이다. PENTIUM Ⅱ의 기억관리장치는 본질적으로인텔의 80386과 80486처리장치에서 사용한것과 같다.

주소공간

PENTIUM Ⅱ는 토막화와 페지화를 위한 하드웨어를 다 포함하고 있다. 사용자가 다음의 기억기에 대한 4가지 독특한 고찰로부터의 선택을 허용하면 두 기구는 리용되지 않을수 있다.

- **비토막화비페지화기억기:** 이 경우 가상주소는 물리주소와 같다. 실례로 이것은 복잡하지 않고 높은 성능을 가진 조종기응용프로그람인 경우에 유효하다.
- **비토막화페지화기억기:** 여기서 기억기는 페지화된 선형주소공간으로 보인다. 기억기의 보호와 관리자페지화를 통해 수행된다. 이것은 일부 조작체계에 의해 실현된다 (실례로 Berkeley UNIX).
- **토막화비페지화기억기:** 여기서 기억기는 론리주소공간들의 집합으로 보인다. 페지화에 비한 우점은 필요하다면 단일바이트준위아래에서의 보호를 가져 온다는것이다. 더우기 페지화와는 달리 토막이 기억기에 있을 때 필요한 변환표가 소편에 있는것을 담보한다. 따라서 토막화비페지화기억기는 접근시간을 예견할수 있다.

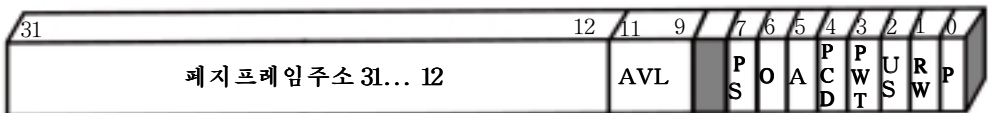
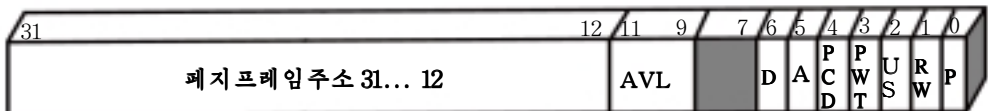
• **토막화페지화기억기:** 토막화는 접근조종의 대상인 론리기억기구획을 정의하는데 사용되고 페지화는 구획안에서의 기억기할당을 관리하는데 사용된다. UNIX 와 같은 조작체계가 이렇게 보인다.

TI－지적자 RPL － 요구특권준위

ㄱ)

ㄴ)

AVL ＝ 체계프로그람에 의한 리용허가
D/ B ＝ 조작크기암시비트
DPL ＝ 서술자특권준위
G ＝ 립자설정
P ＝ 토막존재
S ＝ 서술자형태
▩ ＝예약

ㄷ)

PS ＝ 페지크기
US ＝ 사용자/ 체계관리자
RW ＝ 읽기/ 쓰기
PWT ＝ 동시쓰기
PCD ＝ 캐쉬리용불가능
P＝비트존재
A＝ 접근비트

ㄹ)

31 … 12	11 … 9	7	6	5	4	3	2	1	0
페지프레임주소 31… 12	AVL		D	A	PCD	PWT	US	RW	P

D ＝ 페지변경표식

ㅁ)

그림 7-21. Petium Ⅱ 기억기관리형식들

ㄱ－토막선택부, ㄴ－선형주소, ㄷ－ 토막서술자, ㄹ－ 페지목록항목, ㅁ－ 페지표항목

토막화

토막화가 사용되면 매개 가상주소(PENTIUM Ⅱ문서에서는 론리주소라고 한다.)는 16bit 토막참조와 32bit 편위로 이루어 진다. 토막참조의 2bit 가 보호기구에서 취급되여 14bit 가 토막정의에 사용된다. 따라서 비토막화된 기억기에서 사용하는 가상기억기는 2^{32}=4GB 이다. 토막화기억기에서 총 가상기억공간은 2^{46}=64TB 로 보인다. 물리주소공간은 32bit 주소로서 최대 4GB 의 공간을 가리킨다.

총 가상기억공간은 실지 64TB 보다 더 크다. 이것은 처리장치의 가상주소변환이 어느 처리가 현재 능동인가에 따르기때문이다. 가상주소공간은 2 개 부분으로 나누어 진다. 가상주소공간(8K 토막× 4GB)의 절반은 모든 처리가 공유한 전역부분이고 나머지는 국부이며 매 처리에서 구별된다.

매개 토막에 관련된것이 두가지 형태의 보호인데 그것은 특권준위와 접근속성이다. 최대보호(0 준위)에서 최소보호(3 준위)에까지 4 가지 특권준위가 있다. 자료토막과 관련된 특권준위는 그의 《등급》이며 프로그람토막에 할당된 특권준위는 그의 《허가성》이다. 집행프로그람은 그의 허가성준위가 그 자료토막의 특권준위보다 작거나(높은 특권) 같은(같은 특권) 자료토막만을 접근한다.

장치는 이 특권준위가 어떻게 리용되는가를 지령하지 않는다. 이것은 조작체계의 설계와 실현에 관계된다. 특권준위 1 은 대부분 조작체계에 사용되고 특권준위 0 은 기억기관리보호와 접근조종에 종사하는 조작체계의 작은 부분들에 사용된다. 응용프로그람에는 2 개 준위가 부여된다. 많은 체계들에서 응용프로그람은 준위 3 에 놓이며 준위 2 는 사용하지 않는다. 자기자체의 보호기구들의 실현으로 특수응용프로그람보조체계는 준위 2 의 대리자로 된다. 그의 실례로서는 자료기지관리체계, 사무자동체계, 쏘프트웨어기술환경들이다.

자료토막접근외에 특권기구는 일정한 명령의 사용을 제한한다. 기억관리등록기를 취급하는것 같은 일부 명령들은 오직 준위 6 에서만 집행할수 있다. I/O 명령은 오직 조작체계에 의해 승인된 일정한 준위 즉 일반적으로는 1 준위에서만 집행될수 있다.

자료토막의 접근속성은 읽기-쓰기 또는 읽기만 승인하는가를 가리킨다. 프로그람토막에서의 접근속성이 읽기-수행 또는 읽기전용접근을 지적한다.

토막화에서의 주소변환기는 가상주소를 선형주소로 취급되도록 바꾸는것을 포함한다(그림 7-21 ㄴ). 가상주소는 32bit 편위와 16bit 토막선택부로 이루어 진다(그림 7-21 ㄱ). 토막선택부는 다음의 마당들로 이루어 진다.

- 표지시자(TI): 동토막 또는 국부토막표를 변환에 쓰는가를 가리킨다.
- 토막번호: 막의 번호.이것은 토막표에서의 첨수이다.
- 요구특권준위(RPL): 접근에 요구되는 특권준위

토막표에서 매개 항목은 64bit 로 이루어 져 있다(그림 7-21 ㄷ). 그의 마당들은 표 7-5 에 정의한다.

페지화

토막화는 선택항목이며 불가능할수도 있다. 토막화가 사용될 때 프로그람에서 사용되는 주소는 가상주소이며 이미 설명한대로 선형주소로 변환된다. 토막화를 쓰지 않을 때 선형주소가 프로그람에서 보인다. 다른 경우 선형주소를 실지 32bit 주소로 변환할 때는 다음단계가 리용된다.

선형주소의 구조를 리해하려면 PI 페지화기구가 실지로 그 준위와 조사동작을 한다

는것을 알아야 한다. 첫번째 준위는 1024 항목들로 이루어 진 페지등록부이다. 이것은 4GB 선형주소공간을 1024 개 페지무리로 분리시키는데 한개 크기는 4MB 이다. 매개 페지표는 1024 개 항목으로 이루어 지며 매개 항목은 4kB 페지 하나를 가리킨다. 기억관리는 모든 처리에 한개 페지등록부를 또는 매 처리에 한개 페지등록부를 또는 두가지의 결합의 사용에 대하여 선택한다. 현재과제의 페지등록부는 항상 주기억기에 있다. 페지표는 가상기억기에 있을수 있다.

표 7-5. Pentium II 기억기관리파라메터

토막서술자(토막표항목)

기준

4-Gbit 선형주소공간안에 있는 토막시작주소를 정의한다.

D/B bit

코드토막에서 이것은 D 비트이며 연산수들과 주소화방식들이 16bit 혹은 32bit 라는것을 지적한다.

서술자특권준위 (DPL)

토막서술자에 대응하는 토막특권준위을 구별한다.

립도비트(G)

한계마당이 한바이트 혹은 4kbyte 의 단위로 해석되는가를 지적한다.

한계

토막의 크기를 정의한다. 처리장치는 립도비트에 따라 1byte 로부터 1Mbyte 의 토막크기 한계까지 혹은 4Kbyte 로부터 4Gbyte 의 토막크기한계까지 두개의 방식들중 하나로 한계마당을 해석한다.

S bit

주어 진 토막이 체계토막 혹은 코드 및 자료토막인가를 결정한다.

토막존재비트(P)

무페지화된 체계를 사용한다. 그것은 토막이 주기억기에 존재하는가를 지적한다. 페지화된 체계들에서 이것은 늘 1 로 설정한다.

형

각이한 종류의 토막들사이에서 구별하며 접근속성을 지적한다.

페지등록항목과 페지표항목

접근비트(A)

이 비트는 대응하는 페지에 대한 읽기 혹은 쓰기조작이 일어 날 때 페지표의 두 준위에서 처리장치에 의하여 1 로 설정한다.

(D)

이것은 대응하는 페지에서 쓰기조작이 일어 날 때 1 로 설정한다.

페지프레임주소

존재비트가 1 인 경우 기억기에 페지의 물리주소를 제공한다. 페지프레임이 4K 경내에서 대응되여 아래 12bit 는 0 으로, 웃 20bit 만은 항목에 포함된다.

페지캐쉬불가능비트(PCD)

페지로부터 자료가 캐쉬라는것을 지적한다.

페지크기비트(PS)

페지크기가 4Kbyte 혹은 4Mbyte 라는것을 지적한다.

페지쓰기동시비트(PWT)

쓰기-동시 혹은 쓰기-비동시캐쉬화방책이 대응페지에서 자료가 리용된다는것을 지적한다

존재비트(P)

페지표 혹은 페지가 주기억기에 있다는것을 지적한다.

읽기-쓰기비트(RW)

사용자-준위페지에서 페지가 사용자-준위프로그람인 경우 읽기만의 접근 혹은 읽기-쓰기만의 접근인가를 지적한다.

사용자/체계관리자비트(US)

페지가 조작체계에서만 쓸수 있는가 혹은 조작체계와 응용(사용자준위) 모두에서 쓸수 있는가를 지적한다.

그림 7-21은 페지등록부와 페지표의 양식을 보여 주고 그 마당들은 표 7-5에 정의한다. 접근조종기구는 페지 또는 무리기준에 제공된다.

Pentium II는 또한 변환참조완충기를 사용한다. 완충기는 32개 페지표항목을 기억한다. 페지등록기가 변할 때마다 완충기가 지워 진다.

그림 7-22 는 토막화, 페지화기구의 결합을 보여 준다. 명백히 변환참조완충기와 기억기고속완충기는 보이지 않는다.

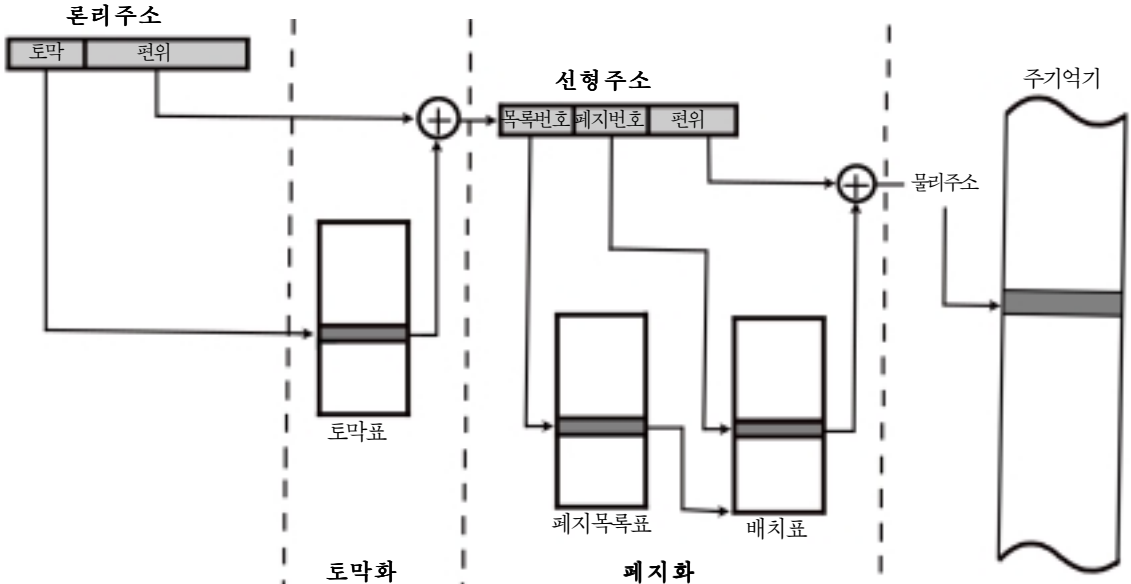



그림 7-22. Pentium II 기억기의 주소변환원리

끝으로 Pentium II는 80386 이나 80486 에는 없는 새로운 확장으로서 2 개 페지를 포함한다. 만일 조종등록기 4 의 PSE(페지크기확장)비트가 1 이면 페지화단위가 조작체계작성자로 하여금 페지를 4KB 나 4MB 크기로 정의하게 한다.

4MB 페지가 사용될 때 페지의 표참조가 한 준위로만 진행된다. 장치가 페지등록부를 접근할 때 페지등록부항목(그림 7 -21 ㄹ)은 PSbit 를 1 로 설정한다. 이 경우 비트 9~21 은 무시되고 비트 22~31 이 기억기의 4MB 페지의 기준주소를 정의한다. 따라서 한개 페지표만이 있다.

4MB 페지를 사용하면 큰 주기억기에 대하여 기억관리등록기요구를 감소시킬수 있다. 4KB 페지주소에 대하여 총 4GB 주기억기는 페지표로서 4MB 기억기를 요구한다. 4MB 페지에서는 4KB 길이이면 페지기억기를 관리하는데 충분하다.

2. Power PC 기억기관리장치

Power PC 는 광범한 주소화기구를 제공한다. 32bit 구성방식의 실현을 위해 간단한 토막화와 페지화기구가 쓰인다. 64bit 실현을 위해서는 페지화와 더 강력한 토막화기구가 지원된다. 더우기 32, 64bit 처리장치에는 블로크주소변환이라는 장치기구가 있다.

페지화에서 수많은 페지들이 프로그람에 의해 자주 참조된다. 실례로 조작체계표나 화면을 완충기로 사용하는 프로그람은 이 동작을 제시한다. 결과는 자주 쓰이는 페지는 계속 페지화된다는것이다. 블로크주소화는 처리장치가 명령기억기와 자료기억기를 크게 4개 블로크로 배치하게 한다.

블로크주소화는 다음장에서 서술한다. 32bit Power PC의 페지화, 토막화기구에 대해서만 본다. 64 bit 방식은 류사한다.

32 bit PowerPC는 32 bit 유효주소를 사용한다(그림 7-23 ㄱ). 주소는 12 bit의 바이트선택부와 16 bit 페지지적자를 포함한다. 따라서 2^{12}=4kbyte 페지가 사용된다. 한 토막당 2^{16}=6K 페지이상이 사용된다. 주소의 4개 비트는 16토막등록기의 하나를 가리키는데 쓰인다. 이 등록기들의 내용은 조작체계가 조종한다. 매개 토막등록기는 조종 bit와

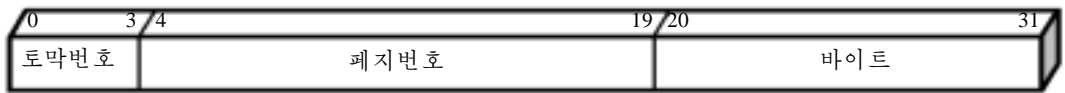
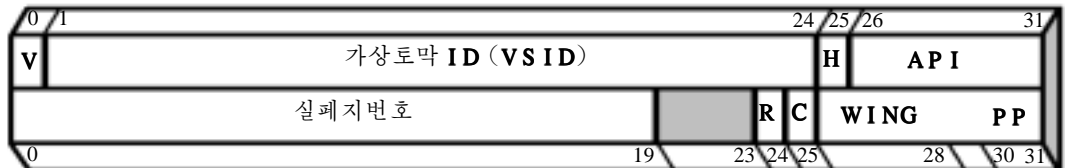
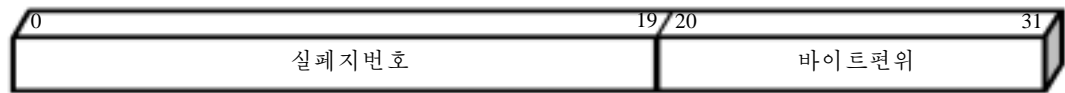



그림 7-23. PowerPC 32bit 기억기관리형식
ㄱ- 유효주소, ㄴ- 페지표항목, ㄷ- 실주소

24bit 지적자를 가지며 그리하여 32bit 유효주소는 52 bit 가상주소로 넘어 간다(그림 7-24).

PowerPC는 하나의 페지표를 사용한다. 가상주소는 다음과 같은 방법으로 페지표에 대한 첨수를 사용한다. 먼저 하쉬코드가 다음과 같이 계산된다.

$$H(0...18)=SID(5...23) \oplus VPN(0...18)$$

가상주소에서 가상페지번호의 왼쪽에 3개조건 0을 덧붙여 19bit수자를 만든다. 그

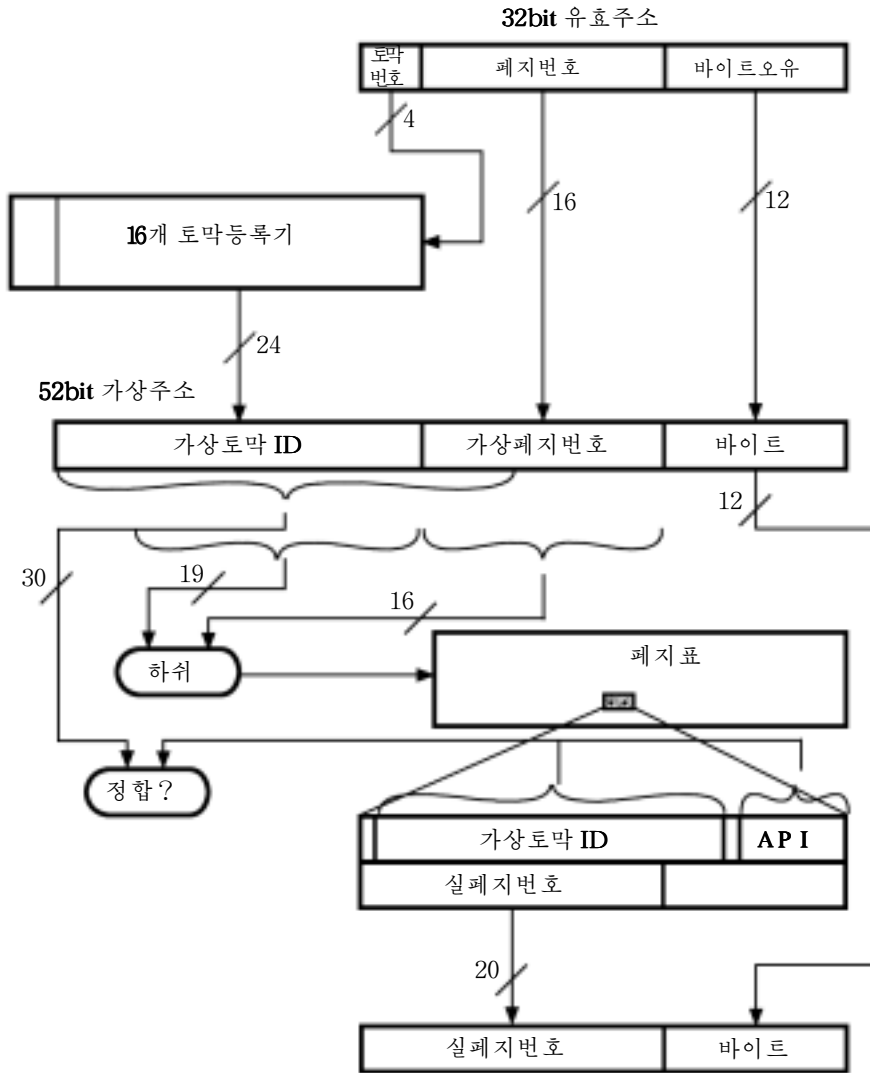



그림 7-24. Powe rPc32bit 주소변환

다음 비트별 배타론리합으로 가상토막 ID의 오른쪽 19bit를 계산하여 19bit 하쉬코드를 형성한다. 표는 8 개 항목의 n 개 무리로 조직된다. 하쉬코드(페지표의 크기에 의존하는)에서 10~19bit는 표에서 무리중 하나를 선택하는데 사용된다. 기억기관리장치는 가상주소와 같은가를 조사하기 위해 항목무리를 조사한다.

그러기 위하여 매개 페지표항목은 가상토막 ID와 비교적 짧은 페지참조인 가상페지번호의 왼쪽 6bit를 포함한다(왜냐하면 16bit 가상페지번호의 10bit는 항상 페지표그룹을 선택하는 하쉬함수에 참가하며 가상페지번호의 단축형만이 가상주소를 조사하기 위하여 페지표에서 수행될수 있기때문이다). 같은것이 있으면 주소에서 실페지번호의

20 bit 가 유효주소의 아래 12bit 와 결합되여 32bit 물리주소를 형성한다.

갈은것이 없으면 하쉬코드는 표의 반대쪽 끝에서 같은 위치에 있는 새 폐지표참조를 만들기 위하여 완성된다. 다음 이 그룹은 적합한 상대에 의하여 조사된다. 같은것이 없으면 폐지오유새치기가 일어 난다.

그림 7-24 는 주소변환기구의 론리를 보여 주며 그림 7-23 은 유효주소, 폐지표항목, 실주소의 양식을 보여 준다. 끝으로 표 7-6 은 폐지표항목에서의 변수들을 정의한다.

64bit 기억기관리방식은 32bit 실현을 개선하도록 설계되였다. 본질적으로 모든 유효주소, 일반등록기, 분기주소등록기들은 64 bit 크기로 왼쪽으로 확장된다.

표 7-6. PowerPC 기억기관리파라메터

토막표기입

유효비트(ID)

64G 유효토막중 한개를 지적한다. 토막표에로의 기입을 결정하는데 리용한다.

기입유효비트(V)

이것이 기억기 혹은 I/O 토막이라는것을 지적한다.

토막종류비트(T)

이것이 기억기 혹은 I/O 토막이라는것을 지적한다

체계관리자 열쇠(Ks)

폐지표에로의 기입을 결정하는데 가상폐지번호를 사용한다.

폐지표기입

기입유효비트(V)

기입에 유효자료가 있다는것을 지적한다.

하쉬기능식별자(H)

이것이 1 차 혹은 2 차하쉬기입이라는것을 지적한다.

간략화된 폐지색인(API)

가상주소를 동일하게 하는데 리용된다.

참조비트(R)

이것은 대응하는 폐지에 대한 읽기, 쓰기조작이 일어 날 때 처리장치에 의하 1 로 설정된다.

변환비트(C)

이것은 대응하는 폐지에 대한 쓰기조작이 일어 날 때 처리장치에 의해 1 로 설정된다.

WIMG 비트들

W=0; 동시쓰기방책사용, W=1; 비동시쓰기방책사용

I=0; 비금지고속완충화, I=1; 금지고속완충화.

M=0; 비공유기억기, M=1; 공유기억기.

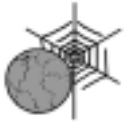
G=0; 비감시기억기, G=1; 감시기억기.

폐지보호비트들(PP)

접근권을 정의하는데 토막등록기 혹은 토막표항목으로부터 K 비트들을 리용한 접근조종비트

참고문헌과 Web 싸이트

[STAL98]은 이장의 문제점들을 자세히 설명한다.

STAL98 Stalling, W. Operating System, Internals and Design Principles, 3rd Edition Upper Saddle River, NJ: Prentice Hall, 1998

Web 싸이트

- **Open Group Research Institute-Operating System Program:** OS 분야의 넓은 범위에서 R & D 를 관리하는 공개그룹(공개쏘프트웨어 지반과 X/Open 회사의 련합)
- **ACM Special Interest Group on Operating System:** SIGOPS 출판물과 협회들의 정보

련습문제

1. 매개 일감이 같은 특성을 가지는 다중프로그람콤퓨터를 가지고 있다고 가정하자. 한 일감에 대한 하나의 계산주기 T 에서 절반시간은 입출력에 돌려 지고 다른 절반은 처리장치동작에 쓴다. 매 일감은 총 N 개 주기에 집행된다. 단순회전순위우선권이 사용되며 입출력동작이 처리장치연산과 겹칠수 있다고 가정하고 다음 지표를 정의하시오.

   - 주기시간=한 일감을 완성하는데 드는 실지시간
   - 처리능력= 주기 T 시간당 완료되는 평균일감수
   - 처리장치리용률=처리장치가 동작(대기아님)중인 시간의 백분률

   이 지표들을 1,2,4 개 일감에 대해 주기 T 가 다음과 같이 분할된다고 가정하고 계산하시오.

   ㄱ)입출력은 첫 절반, 처리장치는 두번째 절반

   ㄴ)입출력은 첫번째와 4 번째의 1/4 주기, 처리장치는 2 번째와 3 번째의 1/4 주기

2. 혼자 집행된다면 한개의 입출력묶음프로그람이 처리장치를 사용한다. 입출력대기시간이 보다 더 길것이다. 처리장치묶음프로그람은 반대이다. 짧은시간일정작성알고리듬이 이미 지나온데서 처리장치시간을 조금 사용한 프로그람을 먼저 출구한다. 이 알고리듬이 입출력묶음프로그람을 먼저 집행시키는가를 설명하시오.

3. 값이 100 인 A 의 100 개렬의 합을 계산하는 식

$$c_i = \sum_{j=1}^{n} a_{ij}$$

을 계산하는 프로그람이 있다. 페지크기가 1000 단어일 때 요구페지법가 발생하며 자료에 할당된 주기억기의 수가 5개 페지프레임이라고 가정하자. A가 가상기억기에 행 또는 렬로 기억될 때 페지오유속도에서 차이가 있는가를 설명하시오.

4. 처리장치에서 현재 집행되는 처리의 페지표가 다음과 같다고 가정하자. 모든 수자는 16 진수이고 모든 수는 0 에서 시작되며 모든 주소는 기억기바이트주소들이다. 페지크기는 1024B 이다.

   ㄱ) 일반적으로 CPU 에서 발생된 가상주소가 물리주기억주소로 어떻게 변환되는가를 정확히 설명하시오.

   ㄴ) 어느 물리주소가 다음가상주소에 해당되는가?

   (1) 1052

   (2) 2221

   (3) 5499

가상페지번호	유효비트	참조비트	변경비트	페지프레임번호
0	1	1	0	4
1	1	1	1	7
2	0	0	0	-
3	1	0	0	2
4	0	0	0	-
5	1	0	1	0

5. 가상기억기체계에서 페지크기가 너무 작지도 않고 너무 크지도 않은 리유는 무엇인가.

6. 다음의 가상페지번호렬은 가상기억기에서 콤퓨터의 집행과정에 부닥친다.

   3 4 2 6 4 7 1 3 2 6 3 5 1 2 3

   최근에 쓰인 페지를 내보내는 방법을 쓴다고 가정하자. $1 \leqq n \leqq 8$ 인 주기억페지용량 n 의 함수로써 페지명중률(주기억기에 있는 페지에 대한 참조률)을 그라프로 그리시오. 주기억기는 초기에 비였다고 가정한다.

7. VAX 콤퓨터에서 사용자페지표는 체계공간의 가상주소에 위치한다. 주기억기보다 가상공간에 사용자페지표를 배치하는것이 어떻게 좋은가? 불리한 점은 무엇인가?

8. 콤퓨터체계가 토막화, 페지화를 가지고 있다고 고찰하자. 토막이 기억기에 있을

때 마지막페지에 일부 단어쪼각이 생긴다. 그외에 토막크기 S, 페지크기가 P 라면 S/P 만한 페지표항목이 있다. 페지크기가 작을수록 토막의 마지막페지에서 쪼각은 작고 페지표는 크다. 종합적인 최소페지크기는 얼마인가?

9. 콤퓨터는 캐쉬, 주기억기, 가상기억기로 쓰이는 디스크를 가진다. 참조된 단어가 캐쉬에 있다면 그것은 접근하는데 20ns 가 필요하다. 주기억에 있으나 캐쉬가 없다면 그것은 캐쉬에 불러 들이는데 60ns 가 필요하며 참조가 다시 시작된다. 만일 주기억기에 없다면 디스크에서 그 단어를 주기억기에 가져 오는데 12ms가 걸리고 캐쉬에 넣는데 60ns 가 걸리며 참조가 계속된다. 캐쉬명중률은 0.9 이고 주기억기명중률은 0.6 이다. 이 체계에서 참조단어를 접근하는데 필요한 평균시간을 ns 로 계산하시오.

10. 한개 과제가 4 개 같은 크기의 토막들로 분할되고 체계가 매개 토막에 8 개 페지서술자표항목을 만든다고 가정하자. 또한 체계는 토막화와 페지화를 결합하여 사용한다. 페지크기는 2KB 라고 가정하자.

ㄱ) 매개 토막의 최대크기는 얼마인가?

ㄴ) 과제의 최대론리주소공간은 얼마인가?

ㄷ) 이 과제가 물리위치 00021ABC 위치의 요소를 접근한다고 가정하자. 과제는 그를 위해 어떤 양식의 론리주소를 형성하는가? 체계의 최대물리주소공간은 얼마인가?

원천 [ALEX93]

11. 극소형처리장치의 물리적주기억기접근공간이 2^{32}B 라고 하자. 그러면 한개 론리토막주소공간은 최대 2^{31}B 이다. 매개 명령은 2 개 부분으로 주소를 가진다. 외부기억관리단위(MMU)가 쓰이는데 그 관리방식은 토막에 고정크기 2^{22}B 의 물리적기억기블로크를 할당한다. 토막의 물리적주소시작번호는 항상 1024 로 나뉘여 진다. MMU 가 어떻게 선택되는가를 설명하시오.

원천 [ALEX93]

12. 페지화된 론리주소공간을 1MB 물리적기억공간으로 넘겼다고 고찰하자.

ㄱ) 처리장치의 론리주소는 어떤 양식인가?

ㄴ) 페지표의 길이와 폭은 얼마인가?

ㄷ) 물리적기억공간이 절반으로 줄 때 페지표에는 어떤 영향이 미치는가?

원천 [ALEX93]

제 3 편. 중앙처리장치

제 3 편의 중심

이제까지는 CPU 를 《검은통》으로 보고 CPU 와 입출력장치(I/O), 기억기 등과의 호상작용을 고찰하였다. 이 편에서는 CPU 의 내부구조와 기능에 대하여 고찰한다. CPU 는 조종장치, 등록기, 산수-론리연산장치, 명령실행장치로 구성되여 있으며 이러한 구성요소들은 서로 결합되여 있다. 제3편에서는 또한 명령모임설계와 자료형과 같은 구조적인 론의와 관흐름조종과 같은 구성과 관련한 내용들을 고찰한다.

제 3 편의 장별 내용

제 8 장. 콤퓨터산수연산

제8장에서는 먼저 산수—론리연산장치의 기능을 고찰하고 그다음 수의 표현와 산수연산의 실현기술을 기본으로 고찰한다. 처리장치들은 일반적으로 두가지 형의 연산 즉 옹근수 혹은 고점수연산과 류점수연산을 진행한다. 이 장에서는 이 두가지 연산에 대하여 우선 수의 표현을 보고 그다음 산수연산을 론의한다. 이 장에서는 국제전기전자기술자협회(IEEE) 754 에서 표준화된 류점수연산을 구체적으로 고찰한다.

제 9 장. 명령모임: 특성과 기능

명령모임설계와 관련한 복잡한 문제들은 제9장과 제10장에서 고찰한다. 제 9 장에서는 명령모임설계의 기능적인 개념들을 기본으로 서술하고 있다. 이 장에서는 우선 콤퓨터명령들로 규정되는 기능형태들을 고찰하고 그다음에 연산에 참가하는 자료를 규정하는 연산수형과 명령모임들에서 일반적으로 찾아 보게 되는 연산종류를 고찰한다.

제 10 장. 명령모임: 주소화방식과 형식

제 9 장에서는 명령모임의 의미를 기본으로 취급하였다면 제10장에서는 명령모임의 문법을 기본으로 고찰한다. 특히 제10장에서는 기억기주소의 규정방법과 콤퓨터명령의 총체적인 형식을 고찰한다.

제 11 장. CPU 구조와 기능

제 11 장에서는 CPU 의 내부기억기인 등록기의 리용에 대하여 서술하고 그다음에 CPU 의 구조와 기능고찰에 필요한 모든 자료들을 보게 된다. 전체 구성(ALU, 조종장치, 등록기파일)을 다시 론의하며 특별히 등록기파일의 구성에 대해서도 론의한다. 또한 기계명령실행에서 처리장치의 기능도 고찰한다. 명령주기는 꺼내기, 간접, 실행, 새치기주기들의 기능과 호상련관성을 보여 주는 방법으로 고찰한다. 마지막으로 성능을 개선하기 위한 관흐름의 리용에 대해서도 깊이 있게 고찰한다.

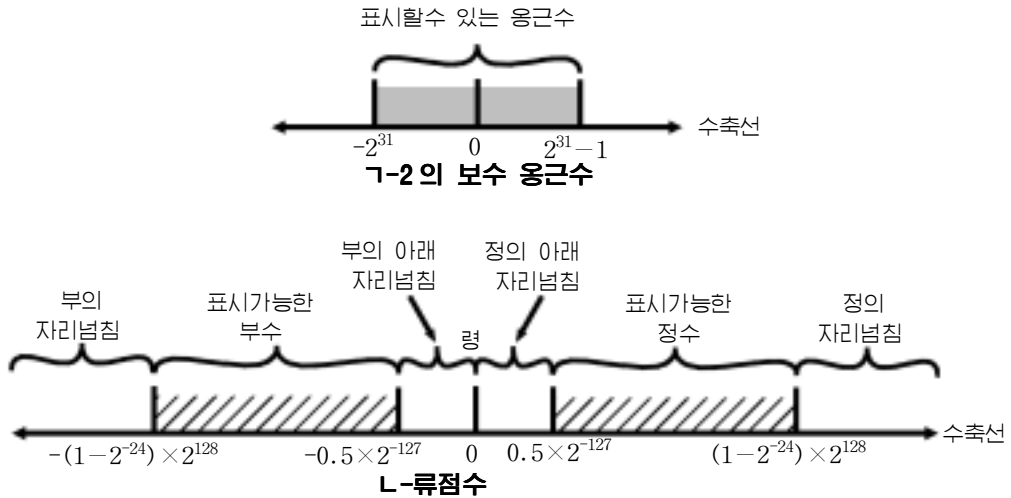
제 12 장. 축소명령모임콤퓨터

제 3 편의 나머지장들에서는 CPU 설계의 기본경향을 보다 상세히 고찰한다. 제 12 장에서는 축소명령모임콤퓨터(RISC)의 구성과 관련한 방법들을 서술하고 RISC 설계를 리용해야 할 필요성들을 고찰한다. 그다음에 RISC 명령모임설계와 RISC CPU 구조를 보다 상세히 고찰한다.

제 13 장. 슈퍼스칼라처리장치

이 장에서는 최근에 처리장치설계에서 많이 리용되는 슈퍼스칼라기술을 고찰한다.

제 8 장. 콤퓨터산수연산

◆◆

- ◆ 콤퓨터산수연산에서 가장 기본적인 두가지 문제는 수를 표현하는 방법(2 진수형식)과 기본산수연산(더하기, 덜기, 곱하기, 나누기)에 리용되는 알고리듬이다. 이 두가지는 옹근수와 류점수산수연산에 다같이 적용된다.
- ◆ 류점수는 어떤 옹근수제곱(지수부)으로 된 상수(밑수)가 곱해 진 수(소수부 혹은 결수부)로 표현된다. 류점수는 매우 크거나 작은 수들을 표현하는데 리용할수 있다.
- ◆ 대다수 처리장치들은 IEEE 754 에서 류점수표현 및 산수연산을 위한 표준화된 권고안에 기초하여 실현한다. IEEE 754 는 32 및 64bit 형식의 류점수표현와 그 연산을 규정하고 있다.

◆◆

이 장에서는 먼저 산수-론리연산장치(ALU)에 대한 총적인 고찰을 한다. 그다음 콤퓨터산수연산과 같은 ALU 에서의 복잡한 개념들에 중점을 두고 론의한다. ALU 가 수행하는 론리기능은 제 9 장에서 고찰하며 간단한 론리회로의 실현과 수자론리회로의 산수기능은 이 책의 부록 A 에 서술한다.

콤퓨터의 산수연산은 일반적으로 완전히 서로 다른 두가지 형의 수들인 옹근수와 류점수에 의하여 진행된다. 어떤 형의 수들을 리용하든지간에 수의 표현법을 선택하는것은 산수연산을 론의함에 있어서 매우 중요한 설계자료로서 우선적으로 취급되여야 한다.

수체계에 대한 구체적인 내용은 이 장의 부록을 참고하면 된다.

제 1 절. 산수 및 론리연산장치

ALU 는 자료의 산수-론리연산을 실지로 수행하는 콤퓨터의 한 부분이다. 조종장치, 등록기, 기억기, 입출력장치(I/O) 와 같은 콤퓨터체계의 다른 모든 구성요소들은 기본적으로 ALU 에 자료를 보내여 거기에서 처리를 한 다음 그 결과들을 가져 간다. 일반적으로 ALU 에 대하여 생각할 때 그것을 콤퓨터의 핵심부로 론의하여 왔다.

콤퓨터에서 ALU 와 모든 전자구성요소들은 2 진수자를 기억하거나 간단한 불(Boolean)론리연산을 수행할수 있는 단순한 수자론리장치들로 구성된다. 수자론리를 실현하는데 흥미가 있는 독자들은 이 장의 부록을 참고하면 된다.

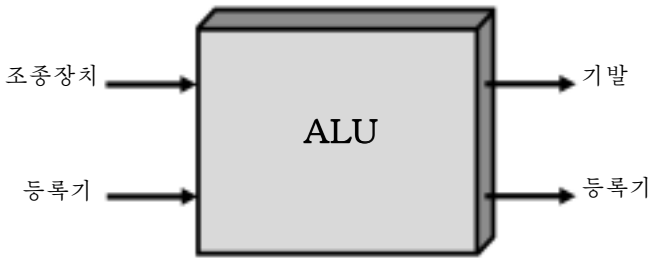



그림 8-1. ALU 의 입구와 출구

그림 8-1 에는 일반적으로 ALU 가 처리장치의 다른 구성요소들과 어떻게 호상 련결되는가를 보여주었다. 자료는 등록기에서 ALU 로 들어 가며 연산결과는 등록기에 기억된다. 이 등록기들은 ALU 에 신호경로로 련결되는 처리장치의 림시기억위치이다(그림 2-3 참고). ALU 는 또한 연산결과로써 기발을 설정하기도 한다. 실례로 자리넘침 기발은 계산결과가 그것을 기억하는 등록기길이를 초과하는 경우에 1 로 설정된다. 기발값은 처리장치내의 등록기들에 기억된다. 조종장치는 ALU 의 연산, ALU 로 들어 오거나 ALU 에서 나가는 자료를 조종하는 신호를 만들어 낸다.

제 2 절. 옹근수표현

2진수체계(부록 8-1 참고)에서 임의의 수는 수자 0과 1, 부호와 주기 혹은 소수점으로 표현할수 있다. 실례를 들면 다음과 같다.

$$-1101.0101_2 = -13.3125_{10}$$

보통 콤퓨터에서는 이 수들을 기억하거나 처리해야 하므로《－》부호와 주기는 리용하지 않는다. 콤퓨터에서는 수를 표현하는데 2 진수자 0 과 1 만을 리용한다. 정의 옹근수만을 리용하는 경우 수의 표현은 매우 간단하다. 8bit 의 단어는 0 부터 255 까지의 수를 표현한다. 실례를 들면 다음과 같다.

00000000 = 0
00000001 = 1
00101001 = 41
10000000 = 128
11111111 = 255

일반적으로 n-bit 의 2 진수자렬 $a_{n-1}a_{n-2}\cdots a_1a_0$ 이 부호 없는 옹근수 A 로 표현된다면 그 값은 다음과 같다.

$$A = \sum_{i=0}^{n-1} 2^i a_i$$

1. 부호－크기표현

정의 옹근수는 물론 부의 옹근수를 표현하는데는 여러가지 대응규칙들이 있는데 이 규칙들에서는 단어에서 맨 웃자리비트(제일 왼쪽에 있는 비트)를 모두 부호비트로서 취한다. 다시말하여 제일 왼쪽에 있는 비트가 0이면 그 수는 정수이고 1이면 부수이다.

부호비트를 리용한 가장 간단한 수표현방법은 부호-크기(sign-magnitude)표현법이다. 이 표현법에서는 n-bit로 된 단어인 경우 제일 왼쪽에 있는 부호비트(n번째 비트)를 제외한 나머지 오른쪽 (n-1) 개의 비트들이 옹근수의 크기를 표현한다. 실례를 들면 다음과 같다.

＋18 ＝ 00010010
－18 ＝ 10010010　　　부호-크기

이것을 일반화하여 표현하면 다음과 같다.

$$\text{부호-크기} \qquad A = \begin{cases} \sum_{i=0}^{n-2} 2^i a_i & , \ a_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i a_i & , \ a_{n-1} = 1 \end{cases} \qquad (8\text{-}1)$$

부호－크기표현법은 여러가지 부족점을 가지고 있다. 하나의 부족점은 필요한 연산을 수행하자면 그 연산에 참가하는 수들의 부호와 크기를 모두 고려해야 한다는것이다. 이와 같은 내용은 제3절에서 구체적으로 론의한다. 다른 하나의 부족점은 0을 두가지로 표현하는것이다.

$$+0_{10} = 00000000$$
$$-0_{10} = 10000000 \quad (\text{부호-크기})$$

이것은 단일한 표현법을 리용할 때보다 0 이 있는 연산을 더 힘들게 하며 따라서 쓰기 불편하다. 0은 콤퓨터에서 연산에 많이 참가한다.

이와 같은 부족점으로부터 부호－크기표현법은 ALU 의 옹근수표현에 드물게 리용된다. 그대신 가장 널리 리용되는 표현법은 다음에 보게 되는 2의 보수표현이다.

2. 2의 보수표현

부호－크기표현법과 마찬가지로 2 의 보수표현은 부호비트로써 맨 웃자리비트를 리용하며 이 비트에 의해 옹근수가 정수인가 부수인가를 쉽게 구별할수 있다. 이 표현법에서는 부수인 경우 부호비트를 제외한 다른 비트들이 변화된다는것이 부호-크기표현법과 차이난다. 표 8-1은 이 절과 다음절에서 보게 되는 2의 보수표현와 그 연산의 기본특성을 보여 주고 있다.

2 의 보수표기법에 대한 다수 취급은 부수를 만드는 규칙에 기본을 두고 있다. 그러나 이 절과 제3절에서 고찰하는 내용들은 2의 보수옹근수들을 2의 보수로 표현된 비트들의 무게합의 형태로 정의하고 있다[DATT93]. 이 내용은 앞에서 고찰된 부호가 없는

표현나 부호-크기표현을 론의하면 명백해 진다. 이와 같이 고찰하면 2 의 보수표기를 리용한 산수연산규칙이 일부 특별한 경우들에는 적용되지 않는다는것을 알수 있다.

표 8-1. 2의 보수표현와 연산의 특성

범 위	-2^{n-1} 로부터 $2^{n-1}-1$ 까지
0의 표현법	하나(단일)
부 정	대응하는 정의 옹근수의 매 비트에 대하여 불보수를 취하고 부호 없는 옹근수로 생각한 결과 비트패턴에 1을 더한다.
비트길이의 확 장	왼쪽에 추가하려는 비트자리를 놓고 원래 부호비트값으로 그것을 채운다.
자리넘침규칙	같은 부호를 가진 두 수가 더해 지는 경우 결과가 원래 더해 지는 두 수의 부호와 반대로 되는 경우에만 자리올림이 발생한다.
덜기규칙	A에서 B를 덜기 위하여 먼저 B에 대한 2의 보수를 구하고 그 다음 그것을 A와 더한다.

이제 2의 보수표현법으로 된 n-bit의 옹근수 A를 생각하자. 이때 A가 정수이면 부호비트 a_{n-1}은 령이다. 나머지비트들은 부호－크기표현법에서와 같은 형태로 수의 크기를 표현한다.

$$A = \sum_{i=0}^{n-2} 2^i a_i \quad , \; A \geqq 0 \text{ 인 경우}$$

수 0은 정수로 고찰되며 따라서 부호비트는 0이고 그의 크기비트들은 모두 0이다. 정의 옹근수의 표현범위는 0(크기비트들이 모두 0)부터 2^{n-1}(크기비트들이 모두1)까지이다. 보다 큰 수는 더 많은 비트들을 리용하여 표현한다.

A가 부수이면 부호비트 a_{n-1}은 1이다. 나머지($n-1$)개의 비트들은 2^{n-1}개의 값들중 임의의 한개를 표현한다. 따라서 부의 옹근수표현범위는 -1부터 -2^{n-1}까지이다. 2의 보수표현에서 부의 옹근수에 대한 비트값할당은 부호 없는 옹근수연산과 류사한 간단한 방법으로 할수 있다. 부호 없는 옹근수표현에서는 비트표현으로부터 옹근수값을 계산하기 위한 가장 큰 무게를 가지는 비트가 $+2^{n-1}$이다. 부호비트를 가진 즉 부호 있는 옹근수표현에서는 가장 큰 무게를 가지는 비트가 -2^{n-1}인 경우에 필요한 연산속성들이 얻어 지도록 옹근수값이 만들어 진다. 이 내용은 제3절에서 구체적으로 고찰한다. 이상과 같은것이 2의 보수표현에서 리용되는 규칙이며 이것을 리용하면 부수인 경우 다음과 같은 표현식을 쓸수 있다.

$$\textbf{2의 보수} \qquad A = -2^{n-1} a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i \qquad (8\text{-}2)$$

정의 옹근수인 경우는 $a_{n-1} = 0$이며 식 8-2에서 $-2^{n-1} a_{n-1} = 0$이다. 결국 식 8-2는 정수와 부수에 대한 2의 보수식을 정의한것이라고 볼수 있다.

2의 보수표현은 그림 8-2에서와 같이 기하학적인 고찰을 통해서도 잘 알수 있다 [BENH92]. 그림에서 매 부분의 웃절반에 있는 원은 수축선의 알맞는 토막을 선택하고 그 점들과 대응시켜 얻은것이다. 원우의 임의의 수에서 생각하면 그 수에 +k를 더하는것(혹은 -k를 더는것)은 시계바늘방향으로 k자리 옮기면 되고 그 수로부터 +k를 더는것(혹은 -k를 더하는것)은 시계바늘과 반대방향으로 k자리 옮기면 된다. 만일 산수연산이

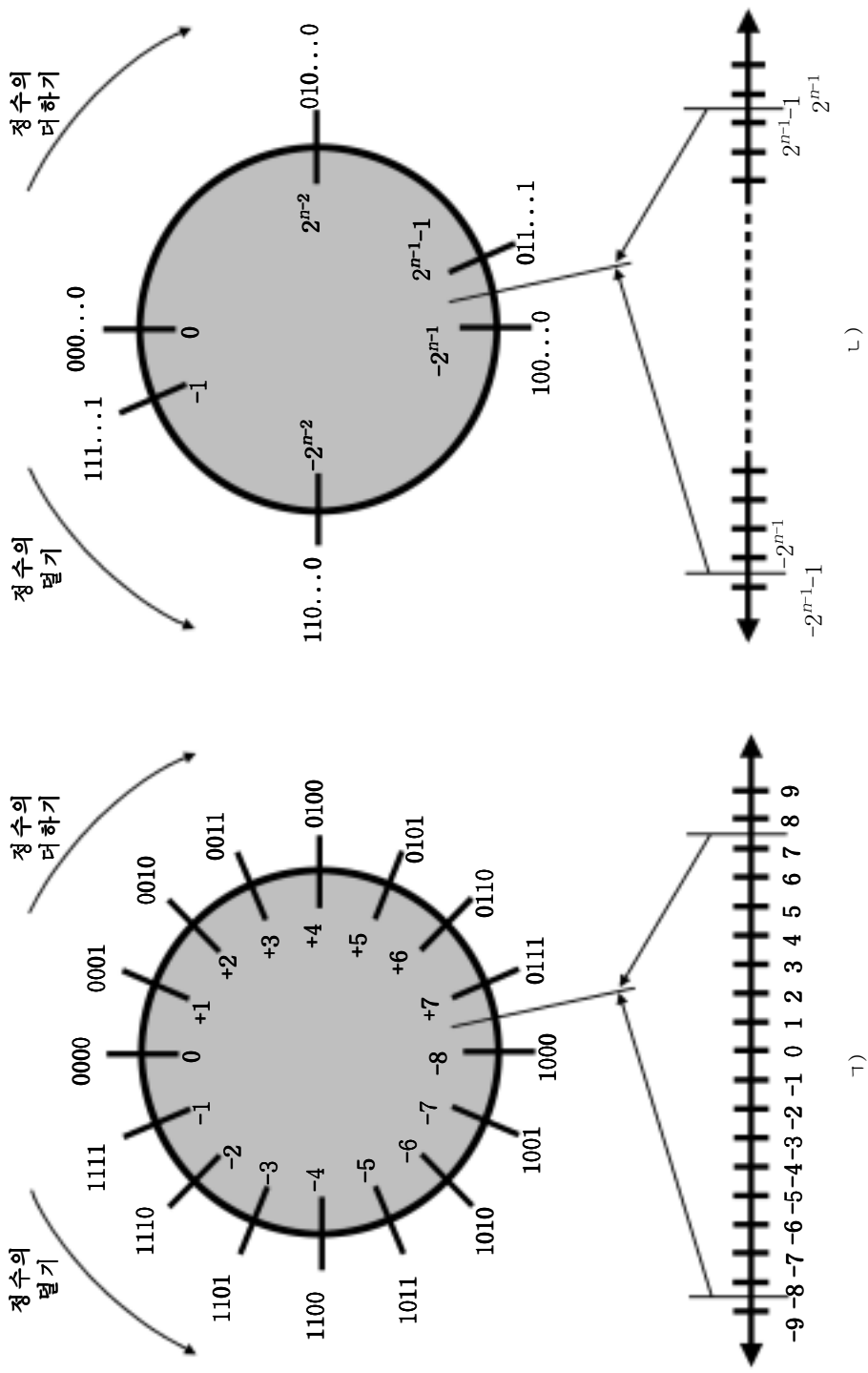



그림 8-2. 2의 보수옹근수의 기하학적고찰

ㄱ-4bit 수인 경우, ㄴ-n-bit 수인 경우

끝점들이 접하는 점을 거쳐 진행되면 정확한 답이 얻어 지지 않는다.

표 8-2에서는 4bit의 옹근수인 경우 부호－크기표현법과 2의 보수표현법을 대비하였다. 2 의 보수표현은 사람이 리해하기에는 어려운 표현법이지만 가장 중요한 산수연산인 더하기와 덜기를 쉽게 하므로 처리장치들에서 옹근수를 표현하는데 가장 널리 리용되고 있다.

표 8-2. 4bit 옹근수들의 서로 다른 표현

10 진표현	부호-크기표현	2 의 보수표현	편위표현
+7	0111	0111	1111
+6	0110	0110	1110
+5	0101	0101	1101
+4	0100	0100	1100
+3	0011	0011	1101
+2	0010	0010	1010
+1	0001	0001	1001
+0	0000	0000	1000
−0	1000	··	0111
−1	1001	1111	0110
−2	1010	1110	0101
−3	1011	1101	0100
−4	1100	1100	0011
−5	1101	1011	0010
−6	1110	1010	0001
−7	1111	1001	0000
−8	··	1000	··

2의 보수표현의 본질에 대한 유익한 설명은 2의 보수값함을 리용해서도 할수 있다. 이 값함은 함에서 제일 오른쪽에 있는 값이 1(2^0)이며 왼쪽에서 그다음위치의 값은 오른쪽 위치에 있는 값의 2배이다. 이렇게 제일 왼쪽에 있는 비트까지 계속되며 제일 왼쪽에 있는 비트는 〈－〉값을 가진다. 그림 8-3 ㄱ에서 볼수 있는바와 같이 표현될수 있는 가장 큰 부의 2 의 보수는 -2^{n-1} 이다. 부호비트를 제외한 임의의 비트들이 1 인 경우 그에 대응하는 수들을 더한다. 또한 부수는 제일 왼쪽에 있는 비트가 1이고 정수는 이 비트가 0 이다. 따라서 가장 큰 정수는 부호비트가 0 이고 나머지비트들은 모두 1 인 수이며 그 값은 $2^{n-1}-1$ 이다.

그림 8-3의 ㄴ와 ㄷ는 2의 보수를 10진수로, 10진수를 2의 보수로 변환하는 값함의 리용을 보여 주고 있다.

3. 각이한 비트길이를 가진 2진수들사이의 변환

우리는 때때로 m < n인 경우 n-bit의 옹근수를 취하여 그것을 m-bit의 옹근수로 기억해야 할 필요성을 느끼게 된다. 부호－크기표기법을 리용하면 이것을 쉽게 실현할수 있다. 즉 부호비트를 새로운 맨 웃자리로 옮기고 그 사이를 0 으로 채우면 된다. 실례를 들면 다음과 같다.

+18 = 00010010 (부호-크기, 8bit)
+18 = 0000000000010010 (부호-크기,16bit)
-18 = 0010010 (부호-크기, 8bit)
+18 = 1000000000010010 (부호-크기,16bit)

이 과정은 부의 옹근수에 대한 2의 보수에서는 적용되지 않는다. 우와 같은 실례를 고찰하면 다음과 같다.

+18 = 00010010 (2의 보수, 8bit)
+18 = 0000000000010010 (2의 보수,16bit)
-18 = 11100110 (2의 보수, 8bit)
+18 = 1000000001101110 (2의 보수,16bit)

마지막행의 전행은 그림 8-3의 값함을 리용하여 쉽게 알수 있으며 마지막행은 식 8-2를 리용하여 증명할수 있다.

-128	64	32	16	8	4	2	1

ㄱ)

-128	64	32	16	8	4	2	1
1	0	0	0	0	0	1	1

-128 +2 +1 = -125

ㄴ)

-128	64	32	16	8	4	2	1
1	0	0	0	1	0	0	0

-120 = -128 +8

ㄷ)

그림 8-3. 2의 보수로 된 2진수와 10진수사이 변환에서 값함의 리용
ㄱ- 8자리 2의 보수값함, ㄴ-2진수 10000011을 10진수로 변환,
ㄷ-10진수 -120을 2진수로 변환

2의 보수표현옹근수인 경우는 부호비트를 새로운 맨 웃자리로 옮기고 그사이를 부호비트로 채우면 된다. 즉 정수이면 0을 채우고 부수이면 1을 채운다. 이것을 부호확장이라고 한다. 이 규칙을 리용하면 －18은 다음과 같이 확장된다.

-18 = 11101110 (2의 보수, 8bit)
-18 = 1111111111101110 (2의 보수, 16bit)

규칙을 해석적으로 고찰하기 위하여 n-bit의 2진수 a^{n-1}을 2의 보수옹근수 A라고 생각하면 그 값은 다음과 같이 표현된다.

$$A = -2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

A 가 정수이면 이 규칙은 명확히 성립한다. 이제 A 가 부수이고 m>n 인 경우 A 를 m-bit 로 확장한다면 그때 A 는 다음과 같이 표현된다.

$$A = -2^{m-1} a_{m-1} + \sum_{i=0}^{m-2} 2^i a_i$$

이 두 값이 반드시 같아야 하므로 다음과 같이 쓸수 있다.

$$-2^{m-1} + \sum_{i=0}^{m-2} 2^i a_i = -2^{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

$$-2^{m-1} + \sum_{i=n-1}^{m-2} 2^i a_i = -2^{n-1}$$

$$2^{n-1} + \sum_{i=n-1}^{m-2} 2^i a_i = 2^{m-1}$$

$$1 + \sum_{i=0}^{n-2} 2^i + \sum_{i=n-1}^{m-2} 2^i a_i = 1 + \sum_{i=0}^{m-2} 2^i$$

$$\sum_{i=n-1}^{m-2} 2^i a_i = \sum_{i=n-1}^{m-2} 2^i$$

$$\Rightarrow a_{m-1} = a_{m-2} = \cdots = a_{n-2} = a_{n-1} = 1$$

식의 전개에서는 첫 식에서 두번째 식으로 가면서 두 표현들에서 맨 아래자리 (n-1)bit 가 변화되지 않도록 식을 써서 마지막식으로부터 두번째 식까지 간다. 마지막식은 (n-1)bit 위치로부터 (m-2)bit 위치까지 모든 비트들이 1 인 경우에만 성립한다. 결국 부호확장규칙은 성립한다.

4. 고점수표현

앞에서 론의한 표현이 때로는 고점수에도 귀착된다. 이것은 소수점(수를 둘로 갈라놓은 점, 간단히 2분할점) 이 고정적으로 맨 아래자리비트의 다음 오른쪽에 있다고 가정하였기때문이다. 프로그람작성자들은 일부 다른 위치로 소수점을 옮겨 수들을 소수점화하여도 앞에서와 같은 표현원리를 리용할수 있다.

제 3 절. 옹근수산수연산

이 절에서는 2 의 보수로 표현된 수들의 일반적인 산수연산기능을 고찰한다.

1. 부 정

부호－크기표현법에서는 옹근수의 부정을 만드는 규칙이 간단하다. 즉 부호비트를 반전시킨다. 2 의 보수표현법에서는 옹근수의 부정이 다음과 같은 규칙에 따라 진행한다.

1. 부호비트를 포함한 옹근수의 매 비트의 불보수를 취한다. 다시말하여 옹근수비트

들에서 1을 0으로, 0을 1로 만든다.

2. 결과를 부호 없는 2진옹근수로 보고 거기에 1을 더한다.

이 두 단계의 처리를 **2의 보수연산** 혹은 옹근수의 2의 보수취하기라고 한다. 실례를 들면 다음과 같다.

```
                +18 =  00010010  (2의 보수)
비트반전보수(1의 보수) =  11101101
                     +        1
                       11101110  = -18
```

2의 보수표현의 원리로부터 2의 보수로 표현된 수의 2의 보수는 그 수자체라는것을 알수 있다. 즉 어떤 수의 부정의 부정은 그 수자체이다.

```
                 -18 = 11101110    (2의 보수)
비트반전보수(1의 보수)  = 00010001
                      +        1
                       00010010 = +18
```

이제 식 8-2에서 보여 준 2의 보수표현의 정의식을 리용하여 2의 보수연산의 정확성을 론증해 보자. 이를 위하여 2의 보수옹근수 A가 n비트의 2진수자렬 $a_{n-1}\,a_{n-2}\cdots a_1\,a_0$으로 되였다고 하면 그 값은 다음과 같이 표현된다.

$$A = -2^{n-1}a_{n-1} + \sum_{i=0}^{n-2} 2^i a_i$$

먼저 매 비트의 부정을 취한 2 진수자렬 $\overline{a_{n-1}}\ \overline{a_{n-2}}\ \cdots\ \overline{a_0}$을 만든다. 이것을 부호 없는 옹근수로 보고 여기에 1을 더한다. 이렇게 얻어 진 n비트의 2진수자렬을 2의 보수옹근수 B라고 하면 그의 값은 다음과 같이 표현된다.

$$B = -2^{n-1}\overline{a_{n-1}} + \sum_{i=0}^{n-2} 2^i \overline{a_i}$$

고찰하려고 하는것은 A = -B이며 이것은 A + B = 0 이라는것을 의미한다. A + B = 0이라는것은 다음과 같이 쉽게 증명된다.

$$\begin{aligned} A+B &= -(a_{n-1} + \overline{a_{n-1}}) + 2^{n-1} + 1 + (\sum_{i=0}^{n-2} 2^i (a_i + \overline{a_i})) \\ &= -2^{n-1} + 1 + (\sum_{i=0}^{n-2} 2^i) \\ &= -2^{n-1} + 1 + (-2^{n-1} - 1) \\ &= -2^{n-1} + 2^{n-1} = 0 \end{aligned}$$

우의 증명에서는 우선 1을 더할 목적으로 옹근수 A의 비트반전보수를 구하고 그것을 부호 없는 옹근수로 가정한 다음 2의 보수옹근수로 취하였다. 이제 두가지 특별한 경우를

고찰해 보자. 먼저 A=0인 경우를 고찰하자. A가 8bit의 길이를 가진다고 하면 다음과 같이 2의 보수가 구해 진다.

0 = 00000000 (2의 보수)
비트반전보수 = 11111111
\+ 1
100000000 = 0

우의 연산에서는 맨 웃자리비트를 초과하는데(우의 연산에서 어두운 부분으로 표현됨)이것은 무시된다. 이로부터 0의 부정은 0이라는것을 알수 있다.

두번째 경우는 n 비트길이의 옹근수에서 맨 웃자리비트만이 1이고 나머지 n-1 개의 비트는 0인 경우이다. 이때에는 같은 수가 얻어 진다. 실례를 들어 8bit의 단어인 경우 다음과 같이 된다.

-128 = 10000000 (2의 보수)
비트반전보수 = 01111111
\+ 1
10000000 = -128

일부 이러한 비정상적인 현상은 피할수 없는것이다. n비트의 단어인 경우 서로 다른 비트렬의 수는 2^n 개로서 짝수이다. 크기가 같은 정 및 부의 옹근수를 부호－크기표현법으로 표현한다면 0을 두가지로 표현할수 있다. 2의 보수표기법으로 표현하는 경우에는 0이 한가지로 표현되며 서로 다른 크기의 부 및 정의 수들을 표현할수 있다. 2의 보수표현법에서는 n비트길이인 경우 -2^n에 대한 표현은 있지만 $+2^n$에 대한 표현은 없다.

2. 더하기와 덜기

2의 보수더하기를 그림 8-4에 보여 주었다. 그림에서 첫 4개의 실례연산들은 정상적인 연산들이다. 연산결과가 정수이면 원래 2 진표기법으로 표현된 정수가 얻어 진다. 연산결과가 부수이면 2의 보수형태로 된 부수가 얻어 진다. 어떤 때에는 단어의 길이를 벗어 나는 비트가 생기는데 이것은 무시한다.

ㄱ)	ㄴ)	ㄷ)
1001 + 0101 1110 = -2	1100 + 0100 10000 = -2	0011 + 0100 0111 = 7

ㄹ)	ㅁ)	ㅂ)
1100 + 1111 11011 = -5	0101 + 0100 1001 = 자리넘침	1001 + 1010 10011 = 자리넘침

그림 8-4. 2의 보수로 표현된 수들의 더하기
ㄱ-(-7)+(+5), ㄴ-(-4)+(+4), ㄷ-(+3)+(+4), ㄹ-(-4)+(-1), ㅁ-(+5)+(+4), ㅂ-(-7)+(6)

더하기에서는 그 결과가 연산에 리용한 단어의 크기보다 더 커질수 있다. 이것을 **자리넘침**(overflow)이라고 한다. 자리넘침이 일어 나면 ALU 는 그 결과를 리용하지 않도록 신호로 표현해야 한다. 자리넘침을 검출하기 위하여 다음과 같은 규칙이 적용된다. 두 수가 모두 정수이거나 혹은 부수인 경우에 더해 지면 자리넘침은 그 결과부호가 더해 지는 두 수의 부호와 반대로 되는 경우에만 발생한다. 그림 8-4 의 ㅁ와 ㅂ가 바로 이러한 자리넘침의 실례로 된다. 자리넘침은 자리올림이 있건 없건 관계없이 일어 날수 있다.

덜기는 다음과 같은 규칙에 따라 쉽게 진행된다. 어떤 수(더는수)를 다른 어떤 수(덜리는수)로부터 덜기 위하여 더는 수의 2 의 보수를 구하고 그것을 덜리는 수와 더한다. 결국 덜기는 그림 8-5에서 보여 준바와 같이 더하기를 리용하여 진행된다. 그림 8-5에서 마지막 두개의 실례는 자리넘침규칙이 여전히 작용한다는것을 보여 준다.

ㄱ)	ㄴ)	ㄷ)
0010 + 1001 1011 = -5	0101 + 1110 10011 = 3	1011 + 1110 11001 = -7

ㄹ)	ㅁ)	ㅂ)
0101 + 0010 0111= 7	0111 + 0111 1110 = 자리넘침	1010 + 1100 10110 = 자리넘침

그림 8-5. 2의 보수로 표현된 수들의 덜기

ㄱ- M = 2 = 0010, S = 7 = 0111, -S = 1001 ,
ㄴ- M = 5 = 0101, S = 2 = 0010, -S = 1110 ,
ㄷ- M = -5 = 1011, S = 2 = 0010, -S = 1110 ,

ㄹ- M = 5 = 0101, S = -2 = 1110, -S = 0010 ,
ㅁ- M = 7 = 0111, S = -7 = 1001, -S = 0111,
ㅂ- M = -6 = 1010, S = 4 = 0100, -S = 1100 ,

그림 8-2에서 고찰해 보면 n비트의 수인 경우 시계바늘이 도는 방향으로 2^n-k위치만큼 옮기면 +k를 덜거나 혹은 -k를 더할수 있다는것을 알수 있다. 그런데 2^n-k는 k에 대한 2의 보수로써 정의한것이다. 이것은 덜기가 더는수의 2의 보수를 더하여 얻어 진다는것을 그라프적으로 보여 주고 있다.

그림 8-6 은 더하기와 덜기연산을 수행하는데 필요한 자료경로와 하드웨어요소들을 보여 주고 있다. 중심적인 요소는 2 진가산기이며 이것은 더하기를 할 두 수들을 더하면 그 합과 자리넘침이 산생한다. 2 진가산기는 두 수들을 부호 없는 옹근수로써 처리한다(가산기의 론리곱하기실현은 부록 I에서 고찰하였다.). 더하기를 해야 할 두수들은 가산기에 들어 간다. 더한 결과는 두 등록기들중 어느 하나에 기억되거나 제3의 다른 등록기에 기억될수도 있다. 자리넘침지시는 한 비트자리넘침기발 (0이면 자리넘침이 없고 1이면 자리넘침이 있다.)에 기억된다. 덜기에서는 더는 수(B 등록기)를 2 의 보수기를 리용하여 2의 보수로 만든 다음 가산기에 넣는다.

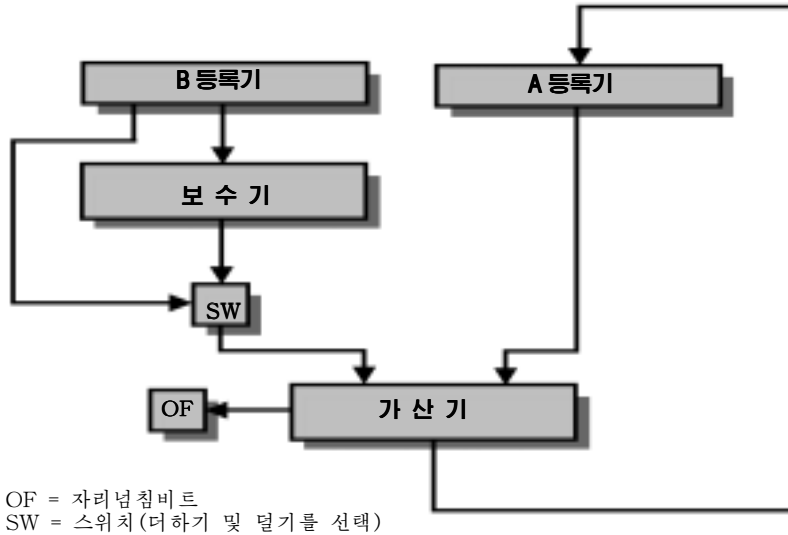



그림 8-6. 더하기 및 덜기의 하드웨어구성도

3. 곱하기

더하기나 덜기에 비하면 곱하기는 그것을 장치적으로 수행하든 프로그람적으로 수행하든 관계없이 복잡한 연산이다. 현재 각종 형태의 곱하기연산알고리듬이 각이한 콤퓨터들에서 리용되고 있다. 이 소제목의 목적은 독자들에게 대표적인 형태의 곱하기방법들을 일정하게 인식시키자는데 있다. 먼저 두개의 부호 없는(부수가 아닌) 옹근수들을 서로 곱하는 간단한 문제를 고찰하고 다음에 2 의 보수로 표현된 수들에서 리용되는 가장 일반적인 곱하기방법들중 하나를 고찰하기로 한다.

부호 없는 옹근수곱하기

그림 8-7 은 종이와 연필을 리용해서 할수 있는 부호 없는 2 진옹근수들의 곱하기를 보여 준다. 곱하기에 대한 몇가지 중요한 고찰들은 다음과 같다.

1. 곱하기는 곱하는수의 매 자리에 대하여 하나씩 부분적이 생긴다. 이 부분적들이 서로 더해 져 최종적인 적이 얻어 진다.
2. 부분적의 정의는 매우 쉽다. 곱하는수의 어떤 비트가 0 이면 부분적이 0 이고 1 이면 부분적은 곱해지는수와 같다.
3. 전체 적은 부분적을 서로 더하여 얻어 진다. 이 연산에서는 매개의 련속되는 부분적이 선행한 부분적에 대하여 상대적으로 왼쪽으로 한자리 밀리워 얻어 진다.
4. 2 개의 n-bit 2 진옹근수의 곱하기는 그 결과가 2n-bit 길이를 가진다.

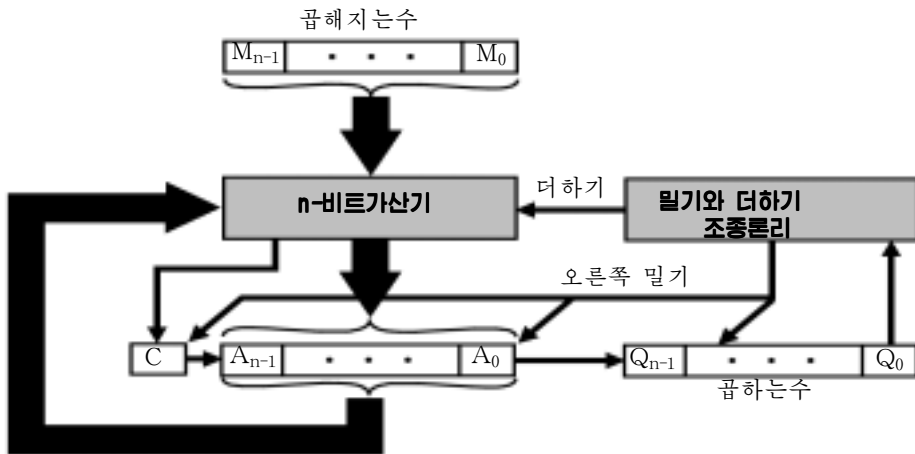
연필을 가지고 종이장우에서 계산하는 방법에 비하여 보다 효과적으로 곱하기연산을 할수 있는 몇가지 방법들이 있다. 우선 부분적이 다 얻어 질 때까지 기다리지 않고 부분적에 대한 계주연산을 진행하는 방법이다. 이 방법은 모든 부분적을 기억하지 않아도 되며 얼마간의 등록기들이 있으면 된다. 다음으로는 부분적을 얻는데서 일부 시간을 절약하는 방법이다. 곱하는수에서 1 인 비트에 대해서는 더하기와 한번의 밀기연산이 진행되

그림 8-7. 부호 없는 2진옹근수의 곱하기

지만 0인 경우에는 밀기연산만 한다.

그림 8-8 ㄱ는 이와 같은 두가지 방법들을 리용한 곱하기실현과정을 보여 주고 있다. 곱하는수와 곱해지는수는 두개의 등록기 Q 와 M 에 각각 넣어 진다. 세번째 등록기인 A등록기는 초기에 0으로 설정된다. 또한 초기에 0 으로 설정되는 한비트의 C 등록기가 있는데 이것은 더하기연산에서 생기는 자리올림비트를 기억하게 된다.

곱하는수의 연산은 다음과 같이 진행된다. 조종론리는 한번에 곱하는수의 하나의 비트만을 읽는다. Q_0이 1이면 곱해지는수는 A 등록기에 더해 지며 결과는 A등록기에 기억된다. 이때 자리넘침이 생기면 C비트에 들어간다. 다음 C, A, Q 등록기들의 모든 비트들이 오른쪽으로 한비트 밀린다. 결과 C는 A_{n-1}에, A_0은 Q_{n-1}에 들어 가고 Q_0은 없어 진다. Q_0이 0이면 더하기연산은 진행되지 않는다.

ㄱ)

C	A	Q	M		
0	0000	1101	1011		초기값
0	1011	1101	1011	Add	첫 주기
0	0101	1110	1011	Shift	
0	0010	1111	1011	Shift	둘째 주기
0	1101	1111	1011	Add	셋째 주기
0	0110	1111	1011	Shift	
1	0001	1111	1011	Add	넷째 주기
0	1000	1111	1011	Shift	

ㄴ)

그림 8-8. 부호 없는 2진수곱하기의 하드웨어실현
ㄱ- 구성도, ㄴ- 그림 8-7로부터의 실례(A,Q의 적)

단지 밀기연산만 진행한다. 이 과정은 곱하는수의 모든 비트에서 반복된다. 2n-bit 의 결과적인 적은 A 와 Q 등록기에 남게 된다. 이와 같은 연산흐름도를 그림 8-9 에 보여 주었으며 그림 8-8 ㄴ에는 그 실례들이 제시되여 있다. 이 흐름도를 통하여 곱하는수의 비트가 0 인 경우에는 더하기연산을 하지 않는다는것을 알수 있다.

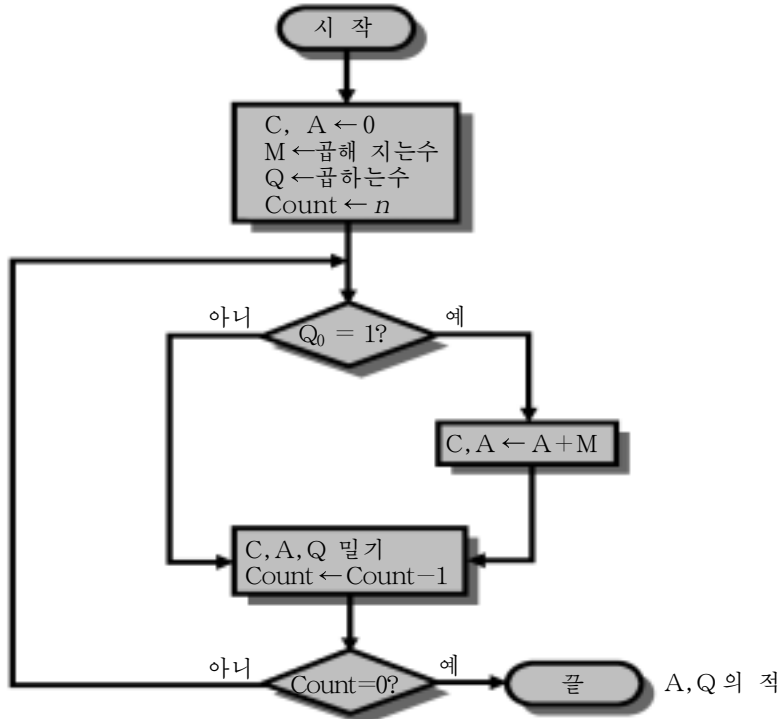



그림 8-9. 부호 없는 2진수곱하기의 흐름도

2 의 보수곱하기

앞에서 고찰한 더하기와 덜기연산에서는 2 의 보수로 표현된 수들을 부호 없는 옹근수로 보고 계산하였다. 이제 다음과 같은것을 생각하자.

$$\begin{array}{r} 1001 \\ +\ \underline{0011} \\ 1100 \end{array}$$

이 연산에서는 두 수들을 부호 없는 옹근수로 보고 9(1001)와 3(0011)을 더하여 12(1100)을 얻는다. 2 의 보수옹근수라고 하면 -7(1101) 에 3(0011) 을 더하여 -4(1100) 를 얻게 된다.

이와 같은 간단한 방법이 곱하기에서는 성립되지 않는다. 이것을 보기 위하여 그림 8-7 을 다시 고찰하자. 그림 8-7 에서는 11(1011)에 13(1101)을 곱하여 143(10001111)을 얻는 과정을 보여 주고 있다. 이 수들을 각각 2 의 보수로 본다면 -5(1011)에 -3(1101)을 곱할 때 -113(10001111)이 된다. 이것은 곱해지는수와 곱하는수가 모두 부수인 경우에는 간단한 곱하기로 결과를 얻을수 없다는것을 명백히 보여 주고 있다. 실지로도 곱하는수와 곱해지는수가 모두 부수인 경우에 우의 연산은 성립되지 않는다. 이 사실을 설명하기

위하여 2의 제곱으로 된 수인 경우 그림 8-7의 연산에서 수행되는것이 무엇인지를 고찰해 보자. 보통 임의의 부호 없는 2진수는 2의 제곱의 합형태로 표현할수 있다. 그러므로

$$1101 = 1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0$$
$$= 2^3 + 2^2 + 2^0$$

2^n의 2 진수곱하기는 그 수를 n-bit 만큼 왼쪽으로 밀어 수행된다. 이것을 리용하면 그림 8-10 에서와 같이 명백한 곱하기의 부분적이 발생하게 된다. 그림 8-10 과의 차이점은 n-bit 의 곱해지는수로부터 산생된 2n-bit 의 수로서 부분적이 고찰되였다는것뿐이다.

```
    1011
   ×1101
--------
00001011     1011×1×2^0
00000000     1011×1×2^1
00101100     1011×1×2^2
01011000     1011×1×2^3
--------
10001111
```

그림 8-10. 8bit 결과를 낳는 두 부호 없는 4bit 옹근수들의 곱하기

이로부터 부호 없는 옹근수로서 4bit 의 곱해지는수 1011 은 8bit 의 단어 00001011 로 기억된다는것을 알수 있다. 매 부분적(2^0에 대한것은 제외)은 왼쪽으로 밀기된 수들로 이루어 진다. 실례로 1011 이 왼쪽으로 두자리 밀리우면 00101100 으로 된다.

이상과 같은 고찰에 기초하면 곱해지는수가 부수인 경우에는 직접적인 곱하기가 성립되지 않는다는것을 알수 있다. 문제는 부분적으로 되는 부의 곱해지는수의 매 값들이 2n-bit 마당에서 반드시 부수여야 한다는것이다. 다시말하여 부분적의 부호비트들은 한줄로 일치되여야 한다. 이것을 그림 8-11을 통하여 고찰해 보자. 이 그림에서는 1001의 곱해지는수에 0011을 곱하는 연산을 각각 보여 주고 있다. 이 수들을 부호 없는 옹근수로 보면 9 × 3 = 27의 곱하기연산은 간단히 진행된다. 그러나 1001을 2의 보수 -7로 보게 되면 그림 8-11 ㄴ 에서 보여 준바와 같이 매 부분적은 2n(8) 비트의 부의 2의 보수로 되여야 한다. 이를 위하여 2진수자 1로 매 부분적의 왼쪽을 채운다.

ㄱ)
```
    1001    (9)
   ×0011    (3)
--------
00001001    1001 × 2^0
00010010    1001 × 2^1
--------
00011011    (27)
```

ㄴ)
```
    1001    (-7)
   ×0011    (3)
--------
11111001    (-7) × 2^0 = (-7)
11110010    (-7) × 2^1 = (-14)
--------
11101011    (-21)
```

그림 8-11. 부호가 없거나 2의 보수로 된 옹근수의 곱하기연산비교
ㄱ-부호 없는 옹근수, ㄴ-2의 보수옹근수

곱하는수가 부수인 경우에도 직접적인 곱하기연산은 성립되지 않는다. 그 리유는 곱하는수의 비트들이 더는 수행해야 할 밀기나 곱하기에 부합되지 않기때문이다. 실례로 10 진수 -3 은 4bit 의 2 의 보수로서는 1101이 된다. 만일 매 비트위치에 기초하여 부분적을 취한다면 다음과 같이 쓸수 있을것이다.

$$1101 \leftrightarrow -(1\times2^3 + 1\times2^2 + 0\times2^1 + 1\times2^0) = -(2^3 + 2^2 + 2^0)$$

실지로 -3은 $-(2^1 + 2^0)$로 표현된다. 이로부터 우에서 서술한 방법에서는 이 곱하는 수를 직접 리용할수 없다. 이것을 해결하기 위한 여러가지 방도들이 있다.

한가지 방법은 곱하는수와 곱해지는수를 모두 정수로 바꾸어 곱하기를 수행하고 곱하기에 참가하는 두 수의 부호가 다른 경우에만 곱하기에 대하여 2 의 보수를 취하는것이다. 그런데 보수기들은 최종 곱하기결과를 변형하지 않는 기술을 더 많이 리용하고 있다. 이러한 방법들가운데서 가장 일반적인 방법의 하나가 바로 부스(Booth)의 알고리듬이다. 이 알고리듬은 보다 직접적인 방법으로 곱하기처리속도를 높인다는 우점을 가지고 있다.

부스의 알고리듬을 그림 8-12 에 보여 주었다. 이 그림은 다음과 같이 설명된다. 앞에서와 같이 곱하는수와 곱해지는수를 특별히 Q 와 M 등록기에 각각 배치한다. 또한 Q 등록기의 맨 아래자리비트 Q_0 의 오른쪽에 배치되는 Q_{-1} 이라고 부르는 1bit 등록기가 있다. 이 등록기의 리용은 뒤에서 간단히 설명된다. 곱하기결과는 A 와 Q 등록기들에 나타난다. A 와 Q_{-1} 은 초기에 0 으로 설정된다. 조종론리는 앞에서와 같이 곱하는수의 비트들을 주사한다. 어느 한 비트가 1 인가 0 인가 결정되면 그의 오른쪽 비트는 다음에 시험된다. 이 두 비트들이 같은 경우에(1 − 1 혹은 0 − 0) A, Q 및 Q_{-1} 등록기들의 모든 비트들이 한 비트 오른쪽으로 밀리우게 된다. 두 비트들이 서로 다른 경우에는 곱해지는 수가 A 등록기에 더해 지거나 A 등록기로부터 덜어 지는데 이것은 두 비트들이 0-1 인가 1-0 인가에 관계된다. 더하기와 덜기에 뒤따라 오른쪽 밀기가 진행된다. 어느 경우에도 오른쪽밀기는 A 의 최고비트 A_{n-2} 로 밀리우거나 A_{n-1}에 그냥 머무르는 연산이다. 이것은 A와 Q에서 수들의 부호를 보존하기 위한것이다. 수들의 부호가 보존되므로 이것을 산수밀기라고 한다.

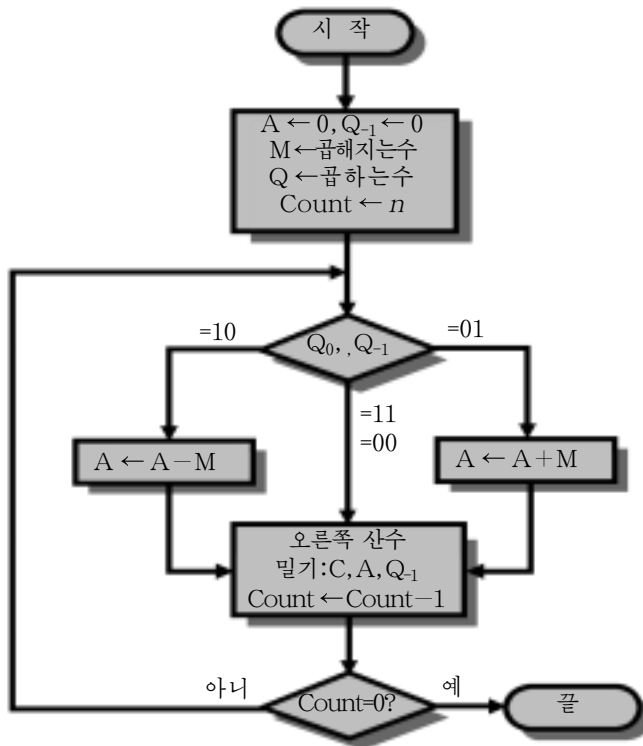



그림 8-12. 2 의 보수곱하기를 위한 부스의 알고리듬

그림 8-13은 7과 3의 곱하기를 위한 부스의 알고리듬에서 사건들이 일어 나는 순서를 보여 주고 있다.

0000	0011	0	0111	초기값	
1001	0011	0	0111	A←A - M	첫 주기
1100	1001	1	0111	밀기	
1110	0100	1	0111	밀기	둘째 주기
0101	0100	1	0111	A←A + M	셋째 주기
0010	1010	0	0111	밀기	
0001	0101	0	0111	밀기	넷째 주기

그림 8-13. 부스의 알고리듬의 실례

그림 8-14에서는 이 연산을 보다 함축하여 보여 주고 있다. 그림 8-14에서 ㄱ를 제외한 나머지부분들은 부스알고리듬의 다른 실례들을 보여 주고 있다. 이 실례들로부터 정수와 부수의 임의의 결합에 대해서도 부스알고리듬이 성립한다는것을 알수 있다. 또한 알고리듬의 효과성도 알수 있다. 1이나 0으로 된 블로크들은 블로크당 한번의 더하기 혹은 덜기만의 평균을 가지고 건너 뛰게 된다.

ㄱ)		ㄴ)		ㄷ)		ㄹ)	
0111		0111		1001		1001	
×0011	(0)	×1101	(0)	×0011	(0)	×1101	(0)
11111001	1-0	11111001	1-0	00000111	1-0	00000111	1-0
0000000	1-1	0000111	0-1	0000000	1-1	1111001	0-1
000111	0-1	111001	1-0	111001	0-1	000111	1-0
00010101	(21)	11101011	(-21)	11101011	(-21)	00010101	(21)

그림 8-14. 부스알고리듬을 리용한 실례

ㄱ- (7)×(3)=(21), ㄴ- (7)×(-3)=(-21), ㄷ- (-7)×(3)=(-21), ㄹ- (-7)×(-3)=(21)

부스알고리듬이 왜 성립하는가? 우선 곱하는수가 정수인 경우를 고찰하자. 이를 위하여 0으로 둘러 막힌 1의 련속렬로 된 블로크(수자렬) 례를 들면 00011110과 같은 정의 곱하는수를 생각하자. 다 아는바와 같이 곱하기는 곱해지는수를 곱하는수에 따라 밀기한 부분적들을 모두 더하여 수행된다.

$$\begin{aligned} M\times(00011110) &= M\times(2^4+2^3+2^2+2^1)= \\ &= M\times(16+8+4+2)= \\ &= M\times30 \end{aligned}$$

연산회수는 다음과 같이 하면 두번으로 감소시킬수 있다.

$$2^n+2^{n-1}+\cdots+2^{n-k}=2^{n+1}-2^{n-k} \qquad (8\text{-}3)$$

따라서

$$\mathrm{M}\times(00011110)=\mathrm{M}\times(2^5-2^1)=$$
$$=\mathrm{M}\times(32-2)=$$
$$=\mathrm{M}\times30$$

그리하여 곱하기결과인 적은 곱해지는수의 한번의 더하기와 한번의 덜기에 의해 얻어 질수 있다. 이 방법은 곱하는수가 모두 1인 비트들로 된 경우를 포함하여 곱하는수에 련속 1인 부분이 있는 임의의 수에로 확장할수 있다. 따라서 다음과 같은 실례를 들수 있다.

$$\mathrm{M}\times(00011110)=\mathrm{M}\times(2^6+2^5+2^4+2^3+2^1)$$
$$=\mathrm{M}\times(2^7-2^3+2^2-2^1)$$

부스알고리듬은 블로크의 첫번째 1이 (1 — 0)과 만나게 되면 덜기를 수행하고 블로크의 끝이 (0 — 1)과 만나게 되면 더하기를 수행하여 이 형식에 부합되게 한다.

이상과 같은 방법이 부의 곱하는수에서도 성립하는가를 보기 위하여 다음과 같은것을 고찰하자. X가 2의 보수로 표현된 부수라고 하면 X는 다음과 같이 표현된다.

$$X\text{의 표현}\qquad X=\{1\mathrm{X}_{n\text{-}2}\mathrm{X}_{n\text{-}3}\ldots\mathrm{X}_1\mathrm{X}_0\}$$

이 X값은 다음과 같이 쓸수도 있다.

$$X=-2^{n-1}+(\mathrm{X}_{n-2}\times2^{n-2})+(\mathrm{X}_{n-3}\times2^{n-3})+\cdots+(\mathrm{X}_1\times2^1)+(\mathrm{X}_0\times2^0)\qquad(8\text{-}4)$$

식 8-4는 표 8-2에 있는 수들에 부스의 알고리듬을 적용하면 얻어 진다.

X의 제일 왼쪽 비트는 X가 부수이므로 1이다. 이제 0이 왼쪽으로부터 k번째의 위치에 있다고 가정하자. 그러면 X는 다음식으로 표현되는 수일수 있다.

$$X\text{의 표현}\qquad X=\{111\ldots10\mathrm{X}_{k\text{-}1}\mathrm{X}_{k\text{-}2}\ldots\mathrm{X}_1\mathrm{X}_0\}\qquad(8\text{-}5)$$

X의 값은 다음과 같이 쓸수 있다.

$$X=-2^{n-1}+2^{n-2}+\ldots+2^{k-1}+(\mathrm{X}_{n-1}\times2^{n-1})+\cdots+(\mathrm{X}_0\times2^0)\qquad(8\text{-}6)$$

식 8-6으로부터 다음과 같은 식을 생각할수 있다.

$$2^{n-2}+2^{n-3}+\ldots+2^{k+1}=2^{n-1}-2^{k+1}$$

웃식을 재배치하면 다음과 같다.

$$-2^{n-2}+2^{n-2}+2^{n-3}+\ldots+2^{k+1}=-2^{k+1}\qquad(8\text{-}7)$$

식 8-7을 식 8-6에 대입하면 다음과 같은 식을 얻을수 있다.

$$X=-2^{k-1}+(\mathrm{X}_{k-1}\times2^{k-1})+\cdots+(\mathrm{X}_0\times2^0)\qquad(8\text{-}8)$$

결국은 부스알고리듬으로 돌아 왔다. X 의 표현[식 8-5]을 상기하면 X_0으로부터 왼쪽에서 제일 첫 0까지의 모든 비트들이 알맞게 조종된다는것이 명백하다. 그것은 비트들이 식 8-8의 (-2^{k+1})을 제외한 모든 항들을 만들며 따라서 적합한 양식으로 된다. 알고

리듬이 왼쪽으로부터 제일 첫 0을 주사하고 그다음 $1(2^{k+1})$과 만나므로 1-0변화가 생기며 덜기가 (-2^{k+1})에서 일어 난다. 이것은 식 8-8 에서 나머지항이다.

실례로 -6 에 곱해지는수를 곱하는 경우를 생각하자. 2 의 보수로 표현된 8bit 단어를 생각하면 -6 은 11111010 으로 표현된다. 식 8-4 를 리용하면 -6 은 다음과 같이 표현된다.

$$-6 = -2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^1$$

따라서

$$M \times (11111010) = M \times (-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^1)$$

웃식에 식 8-7 을 적용하면 된다.

$$M \times (11111010) = M \times (-2^3 + 2^1)$$

이로부터 여전히 M ×(-6)이라는것을 확증할수 있다. 마지막으로 이미전에 증명되여 확증된 방법에 준하면 웃식을 다음과 같이 쓸수 있다.

$$M \times (11111010) = M \times (-2^3 + 2^2 - 2^1)$$

결국 부스알고리듬이 이 형식에 맞는다는것을 알수 있다. 첫 1 이 (1-0)과 만나게 될 때에는 덜기를 하고 (0-1)을 만나게 되면 더하기를 한다. 또한 1 로 된 다음 블로크에서 첫 1 을 만나면 또 다른 덜기를 한다. 그러므로 부스알고리듬은 직접적인 알고리듬보다 더 적은 수의 더하기와 덜기를 수행한다.

4. 나누기

나누기는 곱하기보다 더 복잡하지만 일반적으로 같은 원리에 기초하고 있다. 알고리듬의 기본원리는 곱하기와 마찬가지로 종이장우에서 연필로 하는 방법으로서 이 연산은 반복밀기와 더하기, 덜기를 포함하고 있다.

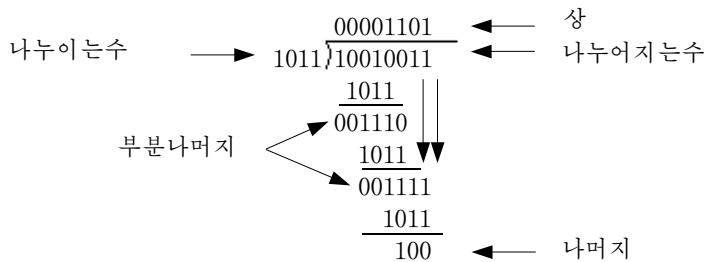



그림 8-15. 부호 없는 2 진옹근수의 나누기

그림 8-15 는 부호 없는 2 진옹근수의 나누기실례를 보여 준다. 우선 나누어지는수의 비트들이 나누는수보다 크거나 같아 지게 될 때까지 왼쪽에서부터 오른쪽으로 가면서 검사된다. 나누어지는수가 나누는수보다 작은 경우에는 0 이 상으로서 왼쪽에서 오른쪽으로 가면서 배치된다. 크거나 같아 지게 되면(이것을 사건이 일어 난다고 한다.) 1 이 상으로서 배치되며 나누는수가 나누어지는수로부터 덜어 진다. 이 결과를 **부분나머지**라고 한다. 이와 같은 과정이 반복된다. 매 주기에서는 나누어지는수

에서 일부 비트들이 추가적으로 부분나머지에 붙게 되는데 이것은 결과가 나누는수보다 크거나 같게 될 때까지 계속된다. 이와 같은 처리는 나누어지는수의 모든 비트들에 다 적용될 때까지 계속된다.

그림 8-16은 긴 나누기처리에 알맞는 기계알고리듬을 보여 준다. 나누는수는 M등록기, 나누어지는수는 Q등록기에 배치된다. 매 단계에서 A와 Q등록기들은 왼쪽으로 한비트 밀리운다. A가 부분나머지[1]를 나누는가 하는것을 결정하기 위하여 A에서 M을 던다. A가 M보다 크면 Q_0은 1이 된다. A가 M보다 작으면 Q_0은 0이 되며 이렇게 되면 이전의 값을 재기억하기 위하여 A에 M을 더하고 그것을 A로 한다. 계수기 count는 감소되고 처리는 n단계까지 계속된다. 마지막에 상이 Q등록기에, 나머지는 A등록기에 남게 된다.

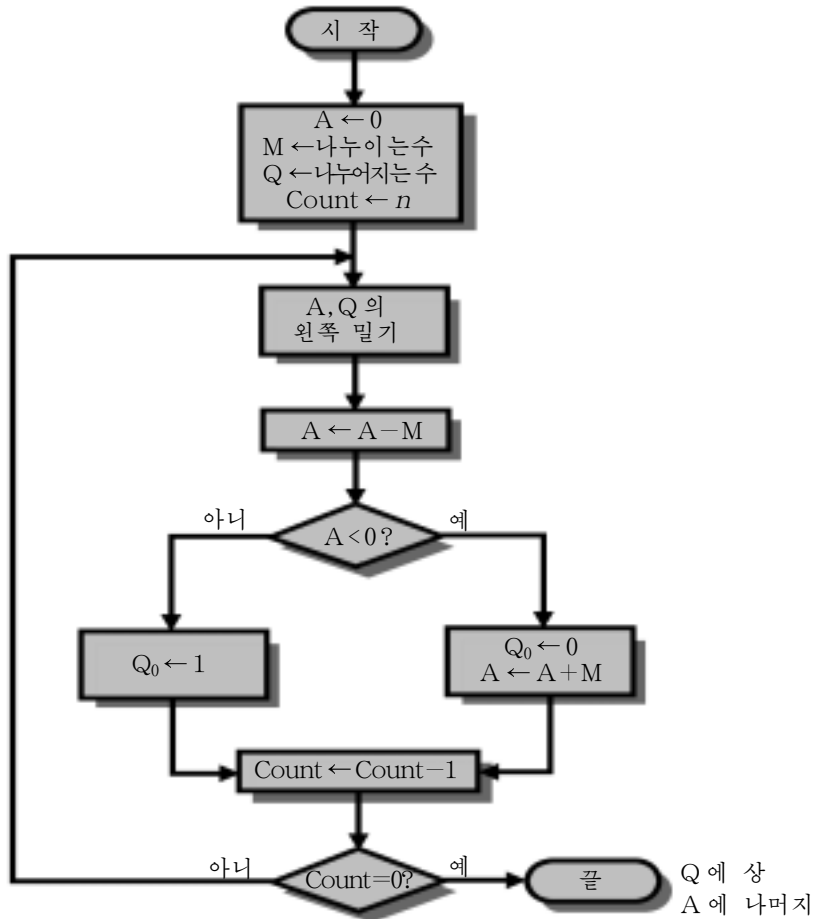



그림 8-16. 부호 없는 2진수나누기흐름도

처리는 어느 정도 복잡하지만 부수인 경우에도 적용할수 있다. 여기서는 2의 보수로 표현된 수들에서 나누기방법의 한가지를 고찰해 본다. 이 고찰방법에 기초한 실례들을 그림 8-17에 보여 주었다. 알고리듬을 요약하면 다음과 같다.

[1] 이것은 부호 없는 옹근수덜기이다. 맨 아래자리비트안으로의 자리내림을 요구하는 결과는 부수이다.

1. M 등록기에 나누이는수를 넣고 A 와 Q 등록기에는 나누어지는수를 넣는다. 나누어지는 수는 2n-bit 의 2 의 보수로 표현되는 수여야 한다. 실례로 4bit 0111 은 00000111 로 되며 1001 은 11111001 로 된다.
2. A 와 Q 의 값을 왼쪽으로 한비트 민다.
3. M 과 A 가 같은 부호이면 A ← A-M 을 수행하고 서로 다른 부호이면 A ← A + M 을 수행한다.
4. 앞의 연산은 A 의 부호가 연산전과 후에 같은 경우 성공으로 된다.
   ㄱ. 연산이 성공이거나 (A=0 AND Q=0)이면 Q_0에 1 을 설정한다.
   ㄴ. 연산이 성공하지 못하여 (A ≠ 0 AND Q ≠ 0) 이면 Q_0에 0 을 설정하고 A 의 이전값을 복귀시킨다.
5. Q 에 비트자리들이 있는 경우에는 그만큼 단계 2 에서부터 단계 4 를 반복한다.
6. 나머지는 A 에 남는다. 나누이는수와 나누어지는수의 부호가 같다면 그때 상은 Q 에 남는다. 이 부호가 다르면 정확한 상은 Q 의 2 의 보수이다.

A	Q	M =0011
0000	0111	초기값
0000	1110	밀기
1101		덜기
0000	1110	복귀
0001	1100	밀기
1110		덜기
0001	1100	복귀
0011	1000	밀기
0000		덜기
0000	1001	Q_0 = 1 설정
0001	0010	밀기

ㄱ)

A	Q	M =1101
0000	0111	초기값
0000	1110	밀기
1101		더하기
0000	1110	복귀
0001	1100	밀기
1110		더하기
0001	1100	복귀
0011	1000	밀기
0000		더하기
0000	1001	Q_0 = 1 설정
0001	0010	밀기

ㄴ)

A	Q	M =0011
1111	1001	초기값
1111	0010	밀기
0010		더하기
1111	0010	복귀
1110	0100	밀기
0001		더하기
1110	0100	복귀
1100	1000	밀기
1111		더하기
1111	1001	Q_0 = 1 설정
1111	0010	밀기

ㄷ)

A	Q	M =1101
1111	1001	초기값
1111	0010	밀기
0010		덜기
1111	0010	복귀
1110	0100	밀기
0001		덜기
1110	0100	복귀
1100	1000	밀기
1111		덜기
1111	1001	Q_0 = 1 설정
1111	0010	밀기

ㄹ)

그림 8-17. 2 의 보수나누기 실례들
ㄱ- (7) ÷ (3), ㄴ- (7) ÷ (-3), ㄷ- (-7) ÷ (3), ㄹ- (-7) ÷ (-3)

그림 8-17 을 보면 (-7) ÷(-3) 의 연산에서 서로 다른 나머지가 나온다. 이것은 나머지가 다음과 같이 정의되기때문이다.

$$D = Q \times V + R$$

여기서 D - 나누어지는수,
Q - 상,
V - 나누이는수,
R - 나머지

그림 8-17 의 결과들은 이 공식과 일치한다.

제 4 절. 류점수표현

1. 원 리

고점수표기법(즉 2 의 보수) 으로는 0 을 중심으로 정 및 부의 옹근수들을 표현할수 있다. 고정된 두 부분 혹은 소수점을 리용한 형식은 소수성분을 가진 수들을 표현할수 있다.

이러한 방법은 일부 제한성을 가진다. 즉 매우 큰 수나 매우 작은 소수들을 표현할수 없다. 특히 두 큰수의 나누기에서 상의 소수부분을 잃어 버릴수 있는 결함이 있다.

10 진수에서는 과학적인 표기법을 리용하여 이 제한성을 극복하고 있다. 976,000,000,000,000 은 9.76×10^{14} 으로 표현할수 있으며 0.000000000000976 은 9.76×10^{-14} 으로 표현할수 있다. 여기서 실지로 알맞는 편리한 위치로 10 진점을 이동하고 그 10 진점의 이동위치를 나타내는 10 의 제곱을 리용한다. 이렇게 하면 매우 큰수나 매우 작은 수를 몇개의 자리수만으로 표현할수 있다.

이와 같은 방법은 2 진수에도 적용할수 있으며 다음과 같은 형태로 2 진수를 표현할수 있다.

$$\pm S \times B^{\pm E}$$

이 수는 세개의 마당을 가진 2 진단어로 기억된다.

- 부호 : ＋ 혹은 －
- 결수 : S
- 지수 : E

밑수 B 는 암시적이며 모든 수들에 공통이므로 기억할 필요가 없다.

2 진류점수의 표현원리는 다음의 실례를 통해서도 잘 알수 있다. 그림 8-18 ㄱ 는 대표적인 32bit 류점수형식을 보여 주고 있다. 맨 웃자리비트는 부호비트로서 0 이면 정수, 1 이면 부수를 나타낸다. 지수값은 그다음 8bit 에 기억된다. 지수표현에서는 편위표현법을 리용한다. 편위라고 하는 고정된 값은 진짜지수값을 가지는 마당으로부터 덜어 진다. 일반적으로 편위는 $(2^{k-1}-1)$과 같은데 여기서 k 는 2 진지수에서 비트들의 개수이다. 이 경우에 8bit 마당은 0 부터 255 까지의 수를 만들므로 127 의 편위로서는 지수값을 -127 ～ +128 의 범위에서 표현할수 있다. 이 실례에서 밑수는 2 이다.

표 8-2 는 4 bit 옹근수에 대한 편위형표현법을 보여 준다. 편위형표현법의 비트들을 부호 없는 옹근수로서 취급하는 경우 수들의 상대적인 크기는 변하지 않는다. 실례로 편위 및 부호 없는 표현에서 가장 큰수는 1111이고 가장 작은수는 0000이다. 이것은 부호-크기, 2의 보수 혹은 1의 보수표현와는 맞지 않는다. 편위형표현의 우점은 부수가 아닌 류점수들을 비교할 목적으로 옹근수로서 취급할수 있다는것이다.

$$
\begin{aligned}
0.11010001 \times 2^{01011} &= 0\ 10010011\ 10100010000000000000000 \\
-0.11010001 \times 2^{01011} &= 1\ 10010011\ 10100010000000000000000 \\
0.11010001 \times 2^{-01011} &= 0\ 10010011\ 10100010000000000000000 \\
-0.11010001 \times 2^{-01011} &= 1\ 10010011\ 10100010000000000000000
\end{aligned}
$$

ㄴ)

그림 8-18. 일반적인 32bit 류점수형식
ㄱ-형식, ㄴ-실례

단어의 마지막부분(그림 8-18 의 경우에는 23bit)은 이른바 소수부라고도 하는 결수부마당이다. 임의의 류점수는 여러가지 형태로 표현할수 있다. 다음의 수들은 결수부를 2 진수형태로 표현한 모두 같은 수들이다.

$$0.110 \times 2^{5}$$
$$110 \times 2^{2}$$
$$0.0110 \times 2^{6}$$

류점수연산을 간단히 하자면 그 수들을 정규화(normalization)해야 한다. 우의 실례에서 정규화된 령이 아닌 수는 다음과 같이 표현할수 있다.

$$\pm\, 0.1bbb \ \ldots\ b \times 2^{\pm E}$$

여기서 b는 2진수자 0혹은 1이다. 이것은 결수부의 제일 왼쪽에 있는 비트가 늘 1이여야 한다는것을 말해 준다. 그러므로 이 비트는 기억할 필요가 없다. 결국 23bit의 마당은 0.5와 1.0사이의 값을 가진 24bit의 결수부를 기억하는데 리용되는것으로 된다.

그림 8-18 ㄴ는 이와 같은 형식으로 된 류점수들의 실례를 보여 주고 있다. 류점수표현의 특징은 다음과 같다.

- 부호는 단어의 첫번째 비트로 기억된다.
- 결수부의 첫 비트는 늘 1이며 따라서 결수부마당에 기억 할 필요가 없다.
- 값 127은 지수부마당에 기억되는 실지 지수에 더해 진다.
- 밑수는 2이다.

그림 8-19 는 이와 같은 표현원리를 리용하여 32bit 단어로 표현할수 있는 수들의 범위를 보여 준다. 2의 보수옹근수표현을 리용하면 $-2^{31} \sim 2^{31}-1$ 까지의 모든 옹근수들을 표현할수 있으며 이것은 총체적으로 2^{32} 개의 서로 다른 수들을 표현할수 있는것으로 된다. 그림 8-18의 실례에서 본 류점수형식에서는 다음과 같은 범위의 수들을 표현할수 있다.

- $-(1-2^{-24}) \times 2^{128}$ 과 -0.5×2^{-127} 사이의 부수
- 0.5×2^{-127} 과 $(1-2^{-24}) \times 2^{128}$ 사이의 정수

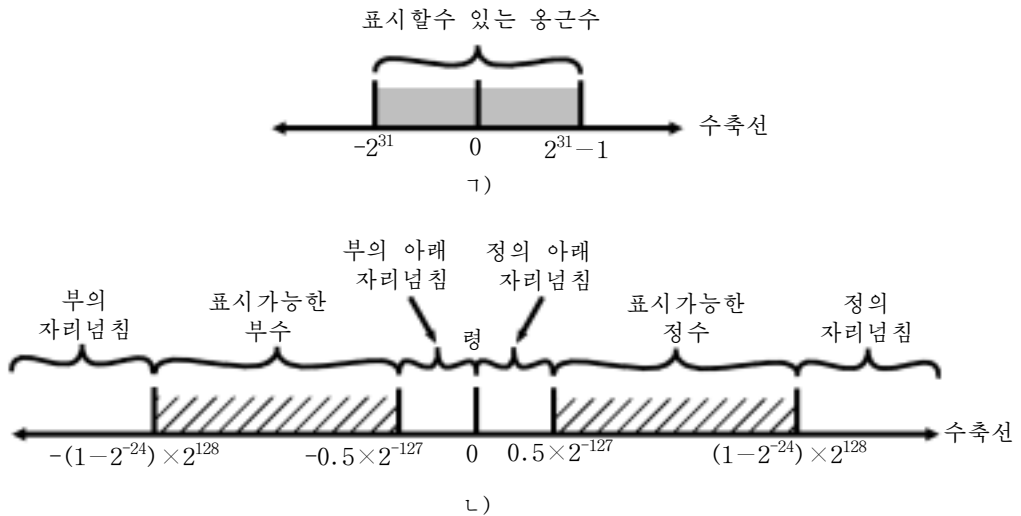



그림 8-19. 32bit 형식으로 표시할수 있는 수
ㄱ-2 의 보수옹근수, ㄴ-류점수

그림 8-19 에서 수축선우에 있은 5 개 령역은 이 범위에 속하지 않는다.

- **부의 자리넘침**이라고 하는 $-(1-2^{-24}) \times 2^{128}$ 보다 작은 부수
- **부의 아래자리넘침**이라고 하는 -0.5×2^{-127} 보다 큰 부수
- **령**
- **정의 아래자리넘침**이라고 하는 0.5×2^{-127} 보다 작은 정수
- **정의 자리넘침**이라고 하는 $(1-2^{-24}) \times 2^{128}$ 보다 큰 정수

이와 같은 표현은 0 에 대해서는 잘 맞지 않는다. 그러나 실지 류점수표현은 령을 가리키는 특별한 비트렬을 가진다. 자리넘침은 산수연산이 128 의 지수로 표현할수 있는 수 $(2^{120} \times 2^{100} = 2^{220})$보다 더 큰 수를 발생시켰을 때 일어 난다. 아래자리넘침은 소수부의 크기가 너무 작을 때 즉 $(2^{-120} \times 2^{-100} = 2^{-220})$보다 작을 때 일어 난다. 아래자리넘침은 그 결과가 0 에 매우 가깝게 다가가는 경우에 생기는 문제이므로 그닥 중요하게 보지는 않는다. 중요한것은 류점수표기에서는 보다 개별적인 값을 표현하고 있지 않다는것이다. 32bit 로 표현할수 있는 각이한 값들의 최대수는 역시 2^{32} 이다. 이미 고찰한것은 정수와 부수로 표현된 두 범위들사이에 수들을 충분히 확산시킨것이다.

또한 류점수표기법으로 표현된 수들은 고점수들과는 달리 수축을 따라 그 간격이 균등하게 놓이지 않는다. 그림 8-20 에서 보여 준것처럼 수값들은 원점에 가까울수록 그 간격이 조밀해 지며 원점에서 멀어 질수록 그 간격은 넓어 진다. 이것은 류점수를 취급하는 수학을 효과적으로 리용할수 있는 전제조건들중의 하나로 된다. 많은 계산들에서는 그 결과가 정확치 않아 가장 가까운 값으로 둥그리기를 하여 다시 표기해야 할 경우가 생긴다.

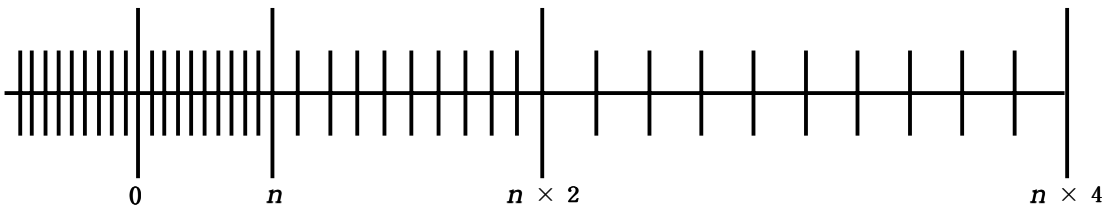



그림 8-20. 류점수의 밀도

그림 8-18에서 보여 준 류점수형식에서 범위와 정확도사이에는 절충방안이 있다. 여기서는 지수표현에 8bit, 결수표현에 23bit 를 할당하고 있다. 만일 지수표현을 위한 비트수를 늘인다면 표현할수 있는 수들의 범위를 확장할수 있다. 그러나 고정된 수의 각이한 값들만이 표현될수 있으므로 이 수들의 밀도는 감소하고 따라서 정확도도 떨어 진다. 범위와 정확도를 둘 다 증가시키는 유일한 방도는 보다 많은 비트를 리용하는것이다. 그리하여 대다수 콤퓨터에서는 적어도 단정확도 및 배정확도수들을 모두 리용한다. 실례로 단정확도형식은 32bit, 배정확도형식은 64bit 이다.

지수표현비트수와 결수표현비트수사이에도 절충방안이 있다. 그러나 이것 역시 문제를 더욱더 복잡하게 만든다. 그리하여 지수의 밑수로 2 를 리용하지 않는 방법을 생각해냈다. 실례로 IBM S/390콤퓨터기본방식에서는 밑수 16을 리용한다. 이 형식은 7bit의 지수부와 24bit 의 결수부로 되여 있다. 이 형식의 실례를 들면 다음과 같다.

$$0.11010001 \times 2^{10100} = 0.11010001 \times 16^{101}$$

이 식으로부터 알수 있는바와 같이 16진수를 리용하면 20이 아니라 5를 지수로 쓸수 있다.

보다 큰 밑수를 리용하면 같은 수의 지수표현비트들을 가지고도 보다 큰 범위에 있는 수들을 표현할수 있다. 그러나 표현할수 있는 각이한 값들의 수는 증가하지 않는다. 결국 고정된 형식에서는 보다 큰 제곱밑수가 보다 작은 정확도로 보다 큰수들의 표현범위를 준다는것을 알수 있다.

2. IEEE 에서 표준화된 2진류점수표현

가장 중요한 류점수표현은 IEEE 표준 754에서 정의되고 있다[IEEE85]. 이 표준은 한 처리장치에서 다른 처리장치에로 프로그람의 이식성(류통성)을 보장하며 고급하면서도 수값처리지향적인 프로그람개발을 촉진할수 있도록 되여 있다. 이 표준은 현재 널리 리용되고 있으며 가상적으로 보면 현 시대의 모든 처리장치들과 산수연산처리장치들에서 리용하고 있다.

IEEE 표준에는 그림 8-21 에서 보여 준바와 같이 지수표현비트수가 각각 8, 11 인 32bit 단정확도형식과 64bit 배정확도형식이 있다. 암시적인 밑수는 2이다. 이외에도 이 표준은 단정확도 및 배정확도를 가지는 2 개의 확장된 형식을 정의하고 있다. 이것들의 정확한 형식은 호상 독립이다. 확장된 형식에는 지수부에 추가적인 비트들(확장범위)과 결수부에 추가적인 비트들(확장정확도)이 있다. 이 확장형식은 중간계산에 리용된다. 확장형식의 정확도가 크면 클수록 지나친 둥그리기오차가 최종결과에 미치는 영향을 작게 할수 있다. 또한 확장형식들의 범위를 크게 하면 계산도중에 자리넘침이 일어 날 가능성이 작아 져 연산결과가 기본형식으로 표현될수 있다. 추가적인 단정확도확장형식은 보다 높

은 정확도를 얻는데 시간을 소비함이 없이 배정확도형식과 같은 우점을 나타내게 한다. 표 8-3 은 이와 같은 4 가지 형식의 특성들을 종합하여 보여 준다.

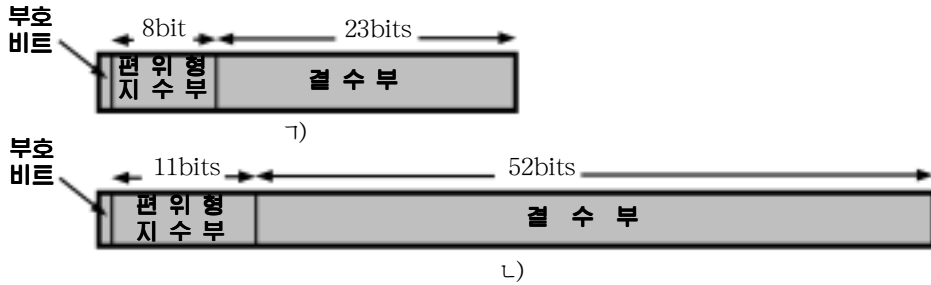



그림 8-21. IEEE 754 형식
ㄱ—단정확도형식, ㄴ—배정확도형식

표 8-3. IEEE 754 형식들의 파라메터

파라메터	단정확도	확장된 단정확도	배정확도	확장된 배정확도
단어비트수(bits)	32	≧ 43	64	≧ 79
지수부비트수(bits)	8	≧ 11	11	≧ 15
지수부 편위	127	미정의	1023	미정의
최대지수부	127	≧ 1023	1023	≧ 16383
최소지수부	-126	≦-1022	-1022	≦-16382
수범위(밑수 10)	$10^{-38}, 10^{+38}$	미정의	$10^{-308}, 10^{+308}$	미정의
결수부비트수(bits)*	23	≧ 31	52	≧ 63
지수부들의 개수	254	미정의	2046	미정의
결수부들의 개수	2^{23}	미정의	2^{52}	미정의
값들의 개수	$1.98*2^{31}$	미정의	$1.99*2^{63}$	미정의

*암시적인 비트는 포함하지 않는다.

IEEE 형식에서 모든 비트렬은 보통 방법으로 해석되지 않는다. 그대신 일부 비트렬들은 특별한 값을 표현하는데 리용한다. 표 8-4 는 여러가지 비트렬에 할당된 값들을 보여 주고 있다. 극단적으로 지수부비트렬에서 모든 비트들이 0 인 값과 모든 비트들이 1(단정확도형식에서는 255, 배정확도형식에서는 2047) 인 값은 특별한 값으로 된다. 류점수로 표현되는 수들의 표현범위와 일부 특이한 수들은 다음과 같다.

- 정규화된 령이 아닌 류점수는 단정확도형식인 경우 지수부가 1～254 범위에, 배정확도형식인 경우 1～2046 의 범위에 있는 수를 표현할수 있다. 지수부는 편위되므로 지수부의 범위는 단정확도형식에서는 -126～+127, 배정확도형식에서는 -1022～ +1023 이다. 정규화된 수에는 소수점의 왼쪽에 1 인 비트가 하나 있어야 한다. 이 비트가 결수부에 포함되여 유효비트수가 단정확도형식에서는 24bit, 배정확도형식에서는 53bit 로 된다.
- 결수부와 지수부가 모두 0 인 수는 부호비트에 따라 +0 혹은 -0 을 표현한다. 이것은 이미 언급된바와 같이 0 의 정확한 값을 표현하는데 효과적으로 리용된다.

- 결수부는 0 이고 지수부의 모든 비트가 1 인 수는 부호비트에 따라 +∞ 혹은 −∞ 를 표현한다. 이 수 역시 무한대를 표현하는데 효과적으로 리용된다.
  이것은 사용자가 오유조건으로써 자리넘침을 취하거나 값을 자리올림하도록 결심할 때까지 그냥 그대로 남는다(프로그람이 실행중에 있어도 진행한다.).
- 결수부는 0 이 아니고 지수부가 0 인 수는 비정규화된 수를 표현한다. 이 경우에 소수점의 왼쪽에 있는 비트는 0 이고 실지 지수부값은 -126 혹은 -1022 이다. 이 수는 부호비트에 따라 + 혹은 −로 된다.
- 결수부는 령이 아니고 지수부의 모든 비트가 1 인 수는 NaN 값을 주는데 NaN(Not a Number)의 의미는 《수가 아니다》라는 뜻이다. 이것은 여러가지 례외조건을 표현하는데 리용한다.

비정규화된 수의 의미와 NaN 에 대해서는 제 5 절에서 고찰한다.

표 8-4. IEEE 754 류점수들의 설명

	단정확도(16bit)				배정확도(64bit)			
	부호	편의된 지 수 부	결수부	값	부호	편의된 지 수 부	결수부	값
+0	0	0	0	0	0	0	0	0
-0	1	0	0	-0	1	0	0	-0
+∞	0	255(모두 1)	0	∞	0	2047(모두 1)	0	∞
−∞	1	255(모두 1)	0	−∞	1	2047(모두 1)	0	−∞
정적 NaN	0 혹은 1	255(모두 1)	≠ 0	NaN	0 혹은 1	2047(모두 1)	≠ 0	NaN
신호화 NaN	0 혹은 1	255(모두 1)	≠ 0	NaN	0 혹은 1	2047(모두 1)	≠ 0	NaN
령아닌 정규화된 정의 수	0	0<e<255	f	$2^{e-127}(1.f)$	0	0<e<2047	f	$2^{e-1023}(1.f)$
령아닌 정규화된 부의 수	1	0<e<255	f	$-2^{e-127}(1.f)$	1	0<e<2047	f	$-2^{e-1023}(1.f)$
비정규화된 정의 수	0	0	f ≠ 0	$2^{e-126}(0.f)$	0	0	f ≠ 0	$2^{e-1022}(0.f)$
비정규화된 부의 수	1	0	f ≠ 0	$-2^{e-126}(0.f)$	1	0	f ≠ 0	$-2^{e-1022}(0.f)$

제 5 절. 류점수산수연산

표 8-5 는 류점수의 기본연산을 개괄하여 보여 주고 있다. 더하기와 덜기에서는 두 연산수들이 같은 지수부값을 가져야 한다는것이 필요조건으로 된다. 이것은 두 연산수들 가운데서 한 연산수의 소수점을 옮겨서 소수점을 일치 혹은 정렬시킬것을 요구한다. 곱하기와 나누기는 보다 더 간단하다.

이와 같은 연산결과 다음과 같은것들이 발생할수 있다.

- **지수부자리넘침 :** 정의 지수부는 최대지수부값을 초과한다. 일부 체계에서는 이것

이 $+\infty$ 혹은 $-\infty$ 로 될수도 있다.

- **지수부아래자리넘침** : 부의 지수부인 경우 최소지수부값보다 작을 때 발생한다. (례 : -200 은 -127 보다 작다) 이것은 그 수가 너무 작아서 표현할수 없다는것을 의미하며 이때에는 0 으로 판단된다.
- **결수부아래자리넘침** : 결수부정렬처리에서는 0 이나 1 의 수자들이 결수부의 오른쪽 끝 밖으로 벗어 날수도 있다. 이때에는 적당한 형태의 둥그리기를 해야 한다.
- **결수부자리넘침** : 같은 부호를 가진 두 결수부의 더하기에서는 맨 웃자리비트 밖으로 자리올림이 생길수 있다. 이것은 재정렬에 의해 고정된다.

표 8-5. 류점수와 산수연산

류점수	산 수 연 산
$X = X_S \times B^{X_E}$ $Y = Y_S \times B^{Y_E}$	$\left.\begin{aligned} X+Y &= (X_S \times B^{X_E - Y_E} + Y_S) \times B^{Y_E} \\ X-Y &= (X_S \times B^{X_E - Y_E} - Y_S) \times B^{Y_E} \end{aligned}\right\}\ X_E \le Y_E$ $X \times Y = (X_S \times Y_S) \times B^{X_E + Y_E}$ $\frac{X}{Y} = \left(\frac{X_S}{Y_S}\right) \times B^{X_E - Y_E}$

실례 :

$X = 0.3 \times 10^{2} = 30$

$Y = 0.2 \times 10^{3} = 200$

$X+Y = (0.3 \times 10^{2-3} + 0.2) \times 10^{3} = 0.23 \times 10^{3} = 230$

$X-Y = (0.3 \times 10^{2-3} - 0.2) \times 10^{3} = (-0.17) \times 10^{3} = -170$

$X+Y = (0.3 \times 0.2) \times 10^{2+3} = 0.06 \times 10^{5} = 6000$

$X+Y = (0.3 \times 0.2) \times 10^{2-3} = 0.23 \times 10^{-1} = 0.15$

1. 더하기와 덜기

류점수의 산수연산인 경우 더하기와 덜기는 곱하기나 나누기보다 더 복잡하다. 그것은 지수부를 반드시 일치시켜야 하기때문이다. 더하기와 덜기알고리듬에는 다음과 같은 4 가지 기본단계가 있다.

1. 령검사
2. 결수부들의 정렬
3. 결수부들의 더하기와 덜기
4. 결과의 정규화

그림 8-22 에는 더하기와 덜기의 일반적인 흐름도를 보여 주었다. 계단식으로 전개된 이 흐름도는 류점수의 더하기와 덜기의 기본기능을 상세히 보여 주고 있다. 류점수의 형식은 그림 8-21 의 류점수형식과 같다. 더하기와 덜기연산에서는 두 연산수가 ALU 에서 리용할수 있는 등록기로 전송되여야 한다. 류점수형식이 암시적인 의미비트를 가지고 있는 경우에는 그 비트가 연산과정에 명시적인것으로 되여야 한다.

더하기와 덜기는 부호변화를 제외하면 같은 연산이므로 그 처리에서는 덜기연산인 경우에만 더는수의 부호를 변화시키고 연산을 시작한다. 또한 두 연산수가운데서 한 연산수가 0 인 경우에는 다른 연산수가 결과로 된다.

그림 8-22. 류점수더하기와 덜기($Z \leftarrow X \pm Y$)

다음단계는 두 지수부가 같아 지도록 수들을 조작하는것이다. 이를 위한 필요조건을 보기 위하여 다음과 같은 10진수더하기를 고찰해 보자.

$$(123 \times 10^{0}) + (456 \times 10^{-2})$$

웃식에서는 명백히 결수부들을 서로 더할수 없다는것을 알수 있다. 결수부를 서로 더하자면 우선 그것의 지수부를 일치시켜야 한다. 구체적으로는 두번째 수의 4가 첫번째 수의 3과 일치되여야 한다. 이렇게 하면 두 지수부들이 같아 지며 바로 이것이 류점수로 표현되는 두수를 더할수 있는 수학적인 필요조건이다. 결국 웃식은 다음과 같다.

$$(123 \times 10^{0}) + (456 \times 10^{-2}) = (123 \times 10^{0}) + (4.56 \times 10^{0}) = 127.56 \times 10^{0}$$

지수부의 정렬은 보다 작은 수는 오른쪽으로(2의 지수부를 증가시킨다.) 보다 큰수는 왼쪽으로 밀기하여 진행한다. 이 연산과정에 수자들이 손실될수 있는데 이 손실은 밀기되는 보다 작은 수들에서 생기게 된다. 다시말하여 연산과정에 손실되는 수자들은 상대적으로 작은 무게를 가진다. 정렬은 결수부 오른쪽 1수자의 크기분을 밀거나 두 지수부가 같아 질 때까지 지수부를 증가시켜 진행한다(밑수가 16인 경우 한 자리의 밀기는 4bit의 밀기와 같다.). 이 정렬처리과정에 결수부가 0이 되면 그때에는 다른 수가 결과로 된다. 결국 두수가 무게적으로 서로 차이나는 지수부를 가지는 경우에는 보다 작은 수가 손실된다.

다음에는 두수의 결수부부호를 고려하여 결수부들이 서로 더해 진다. 부호가 서로 다르면 결과는 0일수 있다. 또한 한 자리의 결수부자리넘침이 생길수도 있다. 결수부자리넘침이 생기면 결수부결과가 오른쪽으로 밀리므로 지수부를 증가시킨다. 이때 지수부자리넘침이 일어 날수 있으며 이것이 체계에 보고되고 연산은 중지된다.

다음단계는 결과를 정규화하는 단계이다. 정규화는 맨 웃자리(밑수16의 지수부인 경우 4bit)가 령이 되지 않을 때까지 결수부를 왼쪽으로 밀어서 진행한다. 밀 때마다 지수부가 감소하며 이렇게 되면 지수부아래자리넘침를 일으킬수 있다. 최종적으로 결과는 둥그리기되여 기록된다. 둥그리기에 대한 론의는 곱하기와 나누기를 고찰한후에 하기로 한다.

2. 곱하기와 나누기

류점수의 곱하기와 나누기는 더하기와 덜기보다 훨씬 더 간단하다.

우선 그림 8-23에 기초하여 곱하기를 먼저 고찰하자. 곱하기에서는 연산수가 0이면 0이 결과로 기록된다. 다음단계는 지수부를 더하는것이다. 지수부가 편위형식으로 기억되면 지수부합은 편위를 2배한것으로 된다. 따라서 편위값을 합에서 덜어야 한다. 이 결과는 알고리듬을 끝내면서 나타나군 하는 지수부자리넘침이나 지수부아래자리넘침을 일으킬수 있다.

다음단계는 적의 지수부가 알맞는 범위내에 있게 되면 두수의 부호를 고려하여 결수부를 곱하는것이다. 곱하기는 옹근수곱하기와 같은 방법으로 수행된다. 이 경우에 여기서는 부호-크기표현로 처리하고 있다. 그러나 내용적으로는 2의 보수표현에서의 곱하기와 류사하다. 곱하기의 적은 곱하는수와 곱해지는수의 길이를 두배한 길이로 될것이다. 범위를 벗어 나는 비트들은 둥그리기하면서 없어 진다.

적을 계산한 다음에는 그 결과를 더하기와 덜기에서처럼 정규화하여 둥그리기한다. 정규화에서는 지수부아래자리넘침이 일어 날수도 있다.

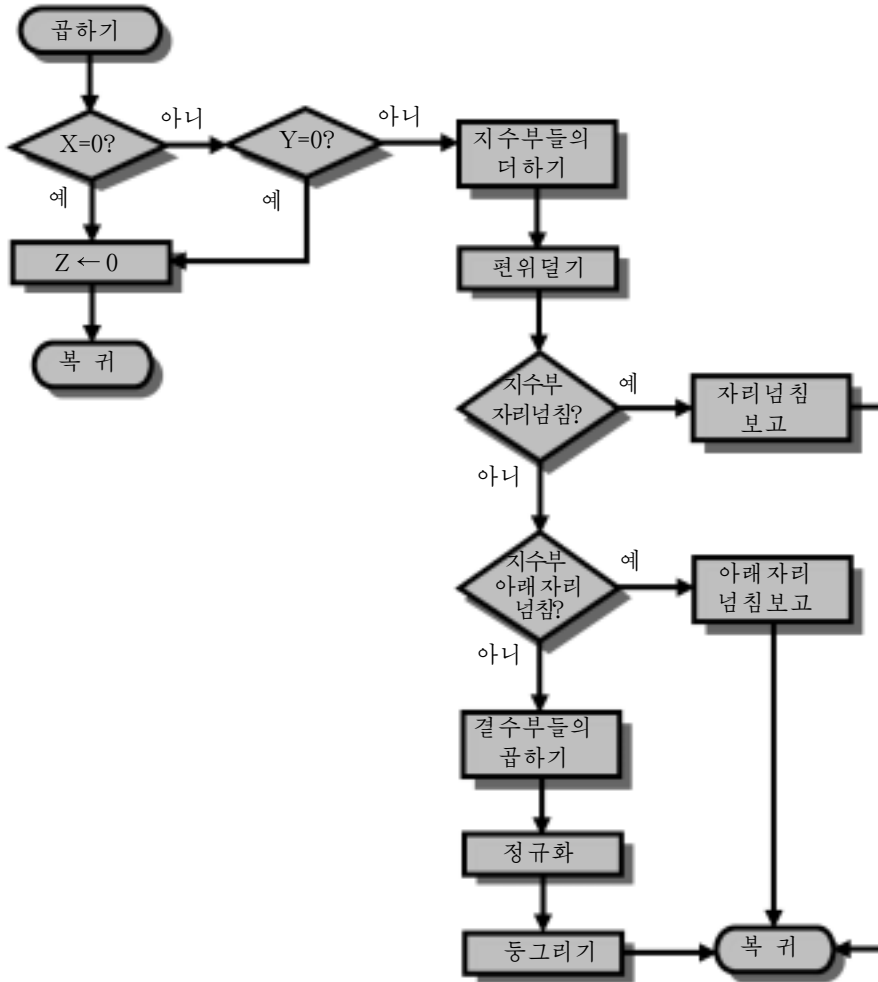



그림 8-23. 류점수들의 곱하기(Z ← X×Y)

마지막으로 그림 8-24를 통하여 나누기과정을 고찰해 보자. 나누기에서도 역시 0에 대한 검사가 첫 단계로 된다. 나누이는수가 0이면 오유통보가 현시되거나 결과가 ∞로 설정된다. 나누어지는수가 0이면 결과는 0이다. 다음단계에서는 나누어지는수에서 나누이는수의 지수부를 던다. 이것은 뒤에서 더해 질 편위를 제거한다. 그리고 지수부아래자리넘침이나 지수부자리넘침이 발생하였는가를 검사한다.

다음단계에서는 결수부를 나누며 이때 보통 정규화와 둥그리기가 뒤따라 진행된다.

3. 정확도고찰

감시비트

앞에서 류점수연산을 하기에 앞서 연산수의 지수부와 결수부가 ALU 등록기에 넣어진다는것을 언급하였다. 결수부의 경우에 그 등록기의 길이는 결수부의 길이에 하나의 암시적인 비트를 더한 길이보다 언제나 더 크다. 이 등록기는 감시비트라고 하는 추가적인 비트들을 가지는데 이것은 결수부의 오른쪽끝을 령으로 채우는데 리용된다.

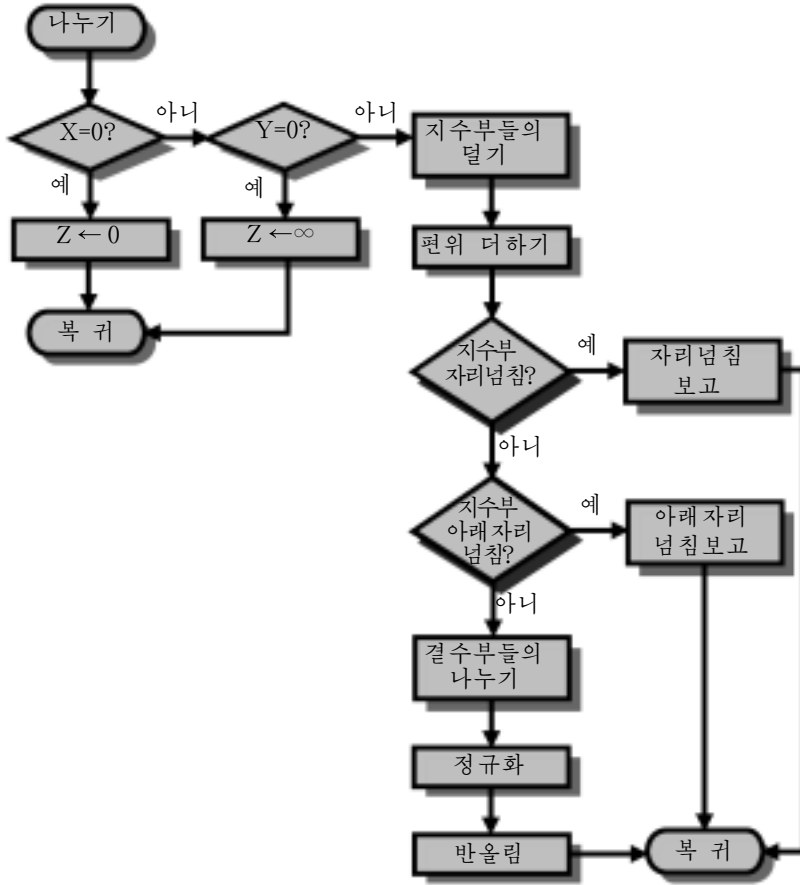



그림 8-24. 류점수들의 나누기(Z←X/Y)

감시비트들을 리용하는 원인을 그림 8-25 에서 설명하고 있다. 2 진소수점의 바로 왼쪽에 2 진수로 1 인 암시비트를 포함하여 총 24bit 로 된 결수부를 가진 IEEE 형식의 수를 고찰하자. 이제 값이 매우 근사한 두수 X=1.00…00 $\times 2^1$ 과 Y=1.11…11 $\times 2^0$ 이 있다고 하자. 작은수를 큰수에서 더는 경우에는 지수부를 맞추기 위하여 한비트를 오른쪽으로 밀어야 한다. 그림 8-25 ㄱ는 이것을 보여 주고 있다. 이 과정에 Y 는 결수부에서 한비트를 잃어 버린다. 그 결과는 2^{-22} 으로 된다. 이와 같은 연산은 감시비트들이 첨가된 그림 8-25 ㄴ에서도 진행된다. 이 경우 맨 아래자리비트는 오른쪽밀기에 의하여 잃어 지지 않으며 그 결과는 그림 8-25 ㄱ에서의 답보다 2 배 작은 2^{-23} 으로 된다. 밑수가 16 인 경우에는 정확도손실이 더 커진다. 그 차이는 그림 8-25 ㄷ와 그림 8-25 ㄹ에서 보여 주고 있는바와 같이 16 의 배수로 된다.

둥그리기

연산결과의 정확도에 영향을 주는 다른 하나의 파라메터는 둥그리기이다. 결수부에 대한 임의의 연산결과는 일반적으로 결수부보다 길이가 긴 등록기들에 기억된다. 연산결과를 류점수형식에 맞추는 경우 범위밖의 초과비트들을 처리해야 한다.

$$
\begin{array}{rl}
x & = 1.000.....00\times 2^{1} \\
\underline{-y} & = \underline{0.111.....11}\times 2^{1} \\
z & = 0.000.....01\times 2^{1} \\
  & = 1.000.....00\times 2^{-22}
\end{array}
\qquad
\begin{array}{rl}
x & = 1.000.....00\ \ 0000\times 2^{1} \\
-y & = \underline{0.111.....11\ \ 1000}\times 2^{1} \\
z & = 0.000.....00\ \ 1000\times 2^{1} \\
  & = 1.000.....00\ \ 0000\times 2^{-23}
\end{array}
$$

ㄱ) ㄴ)

$$
\begin{array}{rl}
x & = .100000\times 16^{1} \\
\underline{-y} & = \underline{.0FFFFF}\times 16^{1} \\
z & = .000001\times 16^{1} \\
  & = .100000\times 16^{-4}
\end{array}
\qquad
\begin{array}{rl}
x & = .100000\ \ 00\times 16^{1} \\
\underline{-y} & = \underline{.0FFFFF\ \ F0}\times 16^{1} \\
z & = .000001\ \ 10\times 16^{1} \\
  & = .100000\ \ 00\times 16^{-5}
\end{array}
$$

ㄷ) ㄹ)

그림 8-25. 감시비트들의 리용
ㄱ- 감시비트가 없는 2진수실례, ㄴ- 감시비트가 있는 2진수실례,
ㄷ- 감시비트가 없는 16진수실례, ㄹ- 감시비트가 있는 16진수실례

이와 같은 둥그리기를 위한 여러가지 방법들이 이미 개발되여 리용되고 있다. IEEE에서도 다음과 같은 4가지 방법들을 표준으로 건의하고 있다.

- **가장 가까운 수로 둥그리기:** 결과를 표현할수 있는 가장 가까운 수로 둥그리기한다.
- **+∞방향으로 둥그리기:** 결과를 +∞방향으로 둥그리기한다.
- **−∞방향으로 둥그리기:** 결과를 −∞방향으로 둥그리기한다.
- **0으로 둥그리기:** 결과를 0으로 둥그리기한다.

이 규칙들을 차례로 고찰해 보자. **가장 가까운 수로 둥그리기**는 IEEE 표준에서 암시적인 둥그리기방식으로써 다음과 같이 둥그리기를 하면 정확한 값에 매우 가깝게 된다. 두개의 가장 가깝게 표현할수 있는 값들이 고르롭게 있다면 그중에서 맨 아래자리비트가 0인 수로 둥그리기를 한다.

실례로 기억해야 할 23개의 비트밖으로 벗어 나는 초과비트들이 10010인 경우 초과비트들은 그 합이 마지막표현비트자리의 절반이상으로 된다. 이 경우에 정확한 답은 다음 표현수에로 올리둥그리기를 하여 마지막표현비트에 2진수 1을 더한것이다. 초과비트들이 01111 인 경우는 초과비트들이 마지막표현비트자리의 절반보다 작게 된다. 이때 그의 정확한 답은 초과비트들을 간단히 잘라 버린것으로써 이것은 다음표현수에 내리둥그리기영향을 미친다. IEEE 표준은 1000… 형식의 특수한 초과비트들에 대해서도 규정하고 있다. 이 경우 결과는 명확히 두 가능한 표현값들의 중간정도로 된다.

이렇게 얻어 진 결과에서 초과비트들을 잘라 버리는것은 가장 간단한 둥그리기연산방법이다. 그러나 이 간단한 방법에도 계산이 진행됨에 따라 작기는 하지만 잘라 버린량이 루적되는 현상이 나타난다.

이것을 해결하기 위한 한가지 방법은 우연수에 기초하여 올리 혹은 내리둥그리기를 하는것인데 이렇게 하면 평균적으로 결과가 한쪽으로만 치우치지 않게 된다. 이 방법의 결함은 예측할수 없으며 또한 확정적인 결과를 얻지 못한다는것이다. 보통 IEEE에서 표준화된 방법들은 결과를 평탄하게 한다. 즉 계산결과가 정확히 두 표현수들사이 중간값이면 마지막표현비트가 1인 경우에 올리둥그리기되며 0인 경우는 내리둥그리기된다.

+∞ **및 −∞방향으로 둥그리기**는 구간산수에 기초하고 있는 방법이다. 구간산수는 모든 결과가 두개의 값을 주므로 류점수계산에서 오유를 감시하거나 조종하는 효과적인 방법으로 리용할수 있다. 구간산수에서 결과로 되는 두값은 실지값을 포함한 구간의 아래끝점과 웃끝점에 해당한다. 구간의 너비는 웃끝점과 아래끝점사이의 차로서 바로 이것

이 둥그리기결과의 정확성을 표현한다. 구간의 끝점들을 표현할수 없는 경우에는 구간끝점들을 특별히 내리 및 올리둥그리기한다. 구간의 너비는 그 실현에 따라 각이하지만 알고리듬은 좁은 구간너비가 얻어 지도록 설계되여 있다. 웃한계와 아래한계사이 범위가 매우 좁은 경우에는 아주 정확한 결과가 얻어 진다. 그렇지 못한 경우 즉 구간너비가 좁지 못한 경우에는 이 내용을 알고 추가적인 해석을 해야 한다.

IEEE 표준에서 규정된 마지막둥그리기방법은 **0 으로 둥그리기**하는것이다. 이것은 사실상 단순한 잘라버리기연산이다. 즉 초과비트들을 무시하는 방법이다. 이 방법은 어느 정도 단순한 벙법으로 보이지만 잘라 버린 값의 크기가 늘 정확한 원래값보다 작거나 같아 지도록 연산에서 0 방향으로 약간 치우치게 결과를 얻어야 한다. 이것은 령이 아닌 초과비트들이 있는 모든 연산에 영향을 주므로 이미 고찰한 둥그리기방법들보다는 더 큰 편의를 가질수 있다.

4. IEEE 에서 표준화된 2 진류점수연산

IEEE 754 는 류점수산수연산이 하드웨어가동환경과는 관계없이 같은 형식이면서도 예측할수 있는 결과를 줄수 있도록 실지 프로그람이나 수속의 바탕으로 되는 간단한 수의 형식만이 아니라 복잡한 수의 형식에 대하여서도 정의하고 있다.

이미 앞에서 고찰한 둥그리기가 바로 여기에 속한다. 이 절에서는 둥그리기를 제외한 다른 세가지 대상들 즉 무한대, NaN 와 비정규화된 수들을 고찰한다.

무한대

무한대산수연산은 실지 산수연산의 극한 경우로 취급되며 모든 수들은 무한대값들사이에 있다.

$$-\infty < (\text{모든 유한인 수}) < +\infty$$

일부 특수한 경우를 제외하고 무한대가 들어 가는 임의의 산수연산은 명백한 결과를 낳는다. 실례를 들면 다음과 같다.

$$
\begin{array}{ll}
5+(+\infty)=+\infty & 5\div(+\infty)=+\infty \\
5-(+\infty)=-\infty & (+\infty)+(+\infty)=+\infty \\
5+(-\infty)=-\infty & (-\infty)+(-\infty)=-\infty \\
5-(-\infty)=+\infty & (-\infty)-(+\infty)=-\infty \\
5\times(+\infty)=+\infty & (+\infty)-(-\infty)=+\infty
\end{array}
$$

정적 및 신호화 NaN

NaN 은 류점수형식으로 부호화된 기호실체로써 여기에는 신호화(signaling) 및 정적(quiet)NaN 의 두가지 형이 있다. 신호화 NaN 은 그것이 연산수로 나타날 때에는 반드시 무효연산레외임을 알린다. 신호화 NaN 은 초기화되지 않은 변수들에 값을 주며 표준화되지 않은 산수연산을 강화하는데 리용된다. 정적 NaN 은 례외(Exception)를 신호함이 없이 거의 모든 산수연산을 진행할수 있게 한다. 표 8-6 은 정적 NaN 을 만드는 연산을 보여 주고 있다.

이 두가지 형의 NaN 은 같은 형식 즉 모두 1 인 지수부와 령이 아닌 결수부를 가진다. 령이 아닌 결수부의 실지 비트렬은 실현독립이다. 이 결수부값에 의하여 신호화 NaN 와 정적 NaN 을 구별하며 또 알맞는 례외조건을 규정할수도 있다.

표 8-6. 정적 NaN 을 만드는 연산

연산	얻어 진 정적 NaN
임 의	신호화 NaN 에서의 임의의 연산
더하기 혹은 덜기	무한대들의 크기 덜기 $(-\infty)+(-\infty)$ $(-\infty)+(+\infty)$ $(+\infty)-(-\infty)$ $(-\infty)-(-\infty)$
곱 하 기	$0 \times \infty$
나 누 기	0/0 혹은 ∞/∞
나 머 지	X 나머지 0 혹은 ∞ 나머지 Y
두 제 곱 뿌 리	$\sqrt{X}$ 여기서 X<0

비정규화된 수

IEEE 754 에서는 지수부 아래자리넘침을 조종하기 위한 비정규화된 수도 규정하고 있다. 류점수의 연산결과 그 지수부가 지내 작으면 즉 매우 큰 부의 지수부라면 그 결과는 결수부를 오른쪽으로 밀기하여 지수부가 표현할수 있는 범위내에 들어 갈 때까지 지수부를 증가시킨 비정규화된 수로 된다.

그림 8-26 은 비정규화된 수의 추가로 인한 영향에 대하여 보여 주고 있다. 보통 표현할수 있는 수는 $[2^n, 2^{n+1}]$의 구간내에 있을수 있다.

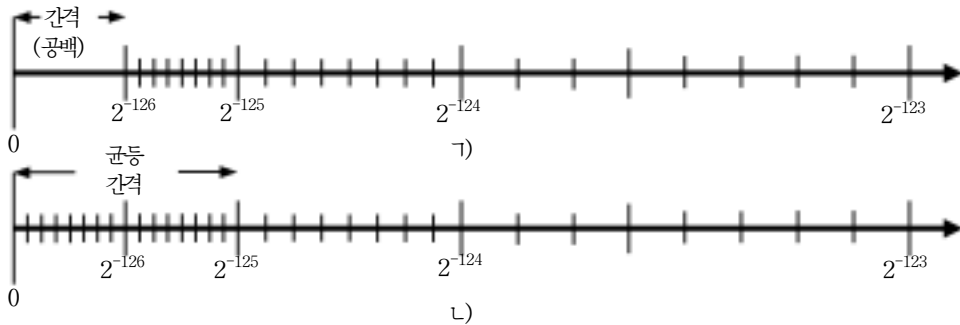



그림 8-26. IEEE 754 비정규화된 수들의 영향
ㄱ—32bit 형식의 정규화된 수,
ㄴ—32bit 형식의 비정규화된 수

매 구간에서 수의 지수부는 그 구간내에서 표현할수 있는 수들의 같은 공간을 조성하므로 결수부가 변하는 동안 일정하게 유지된다. 령으로 다가가면서 고찰해 보면 이때 매 련속되는 구간은 앞구간너비의 절반으로 되지만 같은 개수의 표현할수 있는 수를 가진다. 즉 표현할수 있는 수의 밀도는 령으로 다가갈수록 증가한다. 정규화된 수만을 리용한다면 가장 작은 정규화된 수와 0 사이에 간격이 생긴다. 32bit IEEE 754 형식인 경우 매 구간에는 2^{23} 개의 표현할수 있는 수가 있으며 그의 가장 작은 값은 2^{-126} 이다. 비정규화된 수를 추가하면 2^{23}개의 추가되는 수들이 0 과 2^{-126} 사이에 균등하게 놓이게 된다.

비정규화된 수들은 점차적인 아래자리넘침을 만드는데 리용할수 있다[COON81]. 비정규화된 수들이 없다면 령이 아닌 가장 작은 수와 령사이 간격은 령이 아닌 가장 작은

수와 그다음 보다 큰 수사이의 간격보다 훨씬 더 넓어 진다. 점차적인 아래자리넘침은 이 간격을 채워 줌으로써 정규화된 수들의 둥그리기와 비교될수 있을 정도로 지수부아래자리넘침의 영향을 줄인다.

참고문헌과 Web 싸이트

콤퓨터연산을 배우려는 열성독자들에게는 [SWAR90]의 전 2 권으로 된 책이 좋다. 이 책의 제1권은 1980년에 초판이 출판되였으며 콤퓨터연산의 근본에 대한 론문들이 서술되여 있다. 이 책의 제2권은 콤퓨터연산의 리론, 설계, 실현 등의 측면들을 포괄하여 최근에 발표된 론문들을 담고 있다.

이외에도 콤퓨터연산과 관련한 좋은 책들을 보면 [OMON94], [MORG92] 등이 있다. 류점수연산과 관련하여 잘 알려 진 책은 《모든 콤퓨터과학자는 류점수연산에서 무엇을 알아야 하는가》라는 제목으로 된 [GOLD91] 이다. [WALL90] 은 이 내용을 보다 고급하게 취급하고 있으며 [KNUT98] 은 이에 대한 가장 우수한 책으로서 옹근수연산도 취급하고 있다. 콤퓨터연산과 관련하여 보다 깊이 있게 취급한 가치 있는 책들을 보면 [OBER97a, OBER97b, SODE96] 등이다.

GOLD91 Goldberg, D. "What Every Computer Scientist Should Know About Floationg-Point Arithmetic." *ACM Computiong Surveys*, March 1991. Available at http://www.validgh.com/

KNUT98 Knuth, D. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms.* Reading, MA:Addison-Wesley, 1998

MORG92 Morgan, D. *Numerical Methods*. San Mateo, CA: M&T Books, 1992

OBER97a Oberman, S., and Flynn, M. "Design Issues in Division and Other Floating-Point Operations." *IEEE Transactions on Computers*, February 1997

OBER97b Oberman, S., and Flynn, M. "Division Algorithms and Implementations." *IEEE Transactions on Computers*, August 1997

OMON94 Omondi, A. *Computer Arithmetic Systems:Algorithms, Architecture and Implementations*. Englewood cliffs, NJ: Prentice Hall, 1994

SODE96 Soderquist, P., and Leeser, M."Area and Perfomance Tradeoffs in Floating-Point Divide and Square-Root Implementations." *ACM Computing Surveys*, September 1996

SWAR90 Swartzlander, E., editor, *Computer Arithmetic*, Volumes Ⅰ and Ⅱ. Los Alamitos, CA: IEEE Computer Society Press, 1990

WALL90 Wallis, P. *Improving Floating-Point Programming.* New York: Wiley, 1992

Web 싸이트

- **IEEE 754**: 이 Web 싸이트는 IEEE 754 문서, 관련된 출판물들과 론문들 그리고 콤퓨터산수연산과 관련된 기타 쓸모 있는 관련정보들을 제공한다.

련습문제

1. 2 진옹근수표현에 많이 리용되고 있는 표현법중의 하나는 1 의 보수표현법이다. 이 표현법에서 정의 옹근수는 부호-크기표현법과 같은 방법으로 표현된다. 부의 옹근수는 대응하는 정수의 매 비트에 대하여 불보수를 취하여 표현한다.
   ㄱ. 식 8-1 과 식 8-2 와 류사하게 비트들의 무게합을 리용하여 1 의 보수를 정의하시오.
   ㄴ. 1의 보수로 표현할수 있는 수들의 범위는 얼마인가?
   ㄷ. 1의 보수연산에서 더하기알고리듬을 정의하시오.
2. 표 8-1에 부호-크기표현법과 1의 보수표현법에 대한 렬을 추가하고 전개하시오.
3. 2 진단어에 대한 다음의 연산을 고찰해 보자. 연산은 맨 아래자리비트부터 시작한다.첫 비트가 도착하여 복사될 때까지 0 인 모든 비트들을 복사하시오.그다음 매 비트의 보수를 취하면 결과는 어떻게 되겠는가?
4. 제 3절에서 2의 보수연산을 다음과 같이 정의하였다. X의 2의 보수를 구하자면 X의 매 비트의 불보수를 취하고 거기에 1을 더한다.
   ㄱ. 다음내용이 2의 보수연산에 대한 등가적인 정의라는것을 말해 보시오.
   n비트의 옹근수 X에 대하여 그의 2의 보수는 부호 없는 옹근수로써 X를 취하고 $(2^{n}-X)$를 계산하여 얻는다.
   ㄴ. 그림 8-2 가 ㄱ에서의 요구를 그라프적으로 볼수 있도록 해준다는것을 론증하시오. 이때 시계바늘방향에로의 이동이 어떻게 덜기연산으로 되는가를 보여주시오.
5. 다음의 2의 보수연산에서 차이점을 찾으시오.

ㄱ. 111000	ㄴ. 11001100	ㄷ. 111100001111	ㄹ. 11000011
-110011	-101110	-110011110011	-11101000

6. 아래의 내용이 2 의 보수연산에서 자리넘침에 대한 타당성이 있는 대리정의로 되는가를 말하시오. 제일 왼쪽에 있는 렬의 안쪽과 바깥쪽으로의 자리내림 혹은 자리올림비트들의 안맞음론리더하기가 1 이면 자리넘침조건이 만족되며 그렇지 않으면 자리넘침이 생기지 않는다 .
7. 그림 8-9 와 그림 8-12 를 비교하시오. C 비트는 그림 8-12 에서 왜 리용되지 않는가?
8. 2 의 보수로 표현된 X 와 Y 의 값이 각각 다음과 같이 주어 졌을 때 부스알고리듬으로 적 P = X × Y 를 계산하시요.

$$X = 0101 \qquad (X = 5)$$
$$Y = 1010 \qquad (Y = -6)$$

9. 밑수가 B 인 2 개의 n-자리수들의 곱하기는 2n-자리수보다는 크지 않은 적을 가진다는것을 증명하시오.
10. 그림 8-16 의 부호 없는 2 진수나누기알고리듬의 타당성을 그림 8-15 에서 보여 준 나누기연산단계들로 확증하시오.

11. 제 3 절에서 서술한 2 의 보수옹근수의 나누기알고리듬은 뒤따르는 비성공덜기연산이 A 등록기에서의 값을 회복해야 하므로 회복방법으로 알려 져 있다. 이 방법보다 약간 더 복잡한 방법인 비회복방법은 불필요한 덜기 및 더하기연산을 하지 않는다. 이 방법에 대한 알고리듬을 제기하시오.

12. 콤퓨터의 옹근수연산에서 두 옹근수 J와 K의 상 J/K는 보통 상보다 작거나 같다. 이것이 사실인가?

13. 12 bit 의 단어를 리용하는 2 진 2 의 보수표기로 표현된 수 -145 를 그와 같은 형식의 13 으로 나누시오. 이때 제 3 절에서 서술한 알고리듬을 리용하시오.

14. 지수부 e 는 q 의 편위를 가지고 $0 \leq e \leq x$ 범위에 놓이며 밑수는 b 이고 결수부의 길이가 p 자리라고 하자.

    ㄱ. 프로그람작성에 리용될수 있는 가장 크거나 가장 작은 정수값은 얼마인가?

    ㄴ. 정규화된 류점수로 쓸수 있는 가장 크거나 작은 정수값은 얼마인가?

15. IEEE 32bit 류점수형식으로 다음수들을 표현하시오.

ㄱ. -5	ㄷ. -1.5	ㅁ. 1/16
ㄴ. -6	ㄹ. 384	ㅂ. -1/32

16. 밑수가 16 이고 지수부가 7bit 인 IBM 의 32bit 류점수형식으로 다음수들을 표현하시오.

ㄱ. 1.0 ㄴ. 0.5	ㄷ. 1/64 ㄹ. 0.0	ㅁ. –15.0 ㅂ. 5.4×10^{-79}	ㅅ. 7.2×10^{75}

17. 다음의 경우에 편위값은 얼마인가?

    ㄱ. 밑수 2, 6bit 마당의 지수부

    ㄴ. 밑수 8, 7bit 마당의 지수부

18. 그림 8-21 ㄴ의 류점수형식에 대하여 그림 8-19 ㄴ와 류사한 수축을 그리시오.

19. 8bit 의 편위형지수부와 23bit 의 결수부를 가진 류점수형식이 있다. 다음의 수들에 대한 비트렬을 이 형식으로 보여 주시오.

    ㄱ. -720

    ㄴ. 0.645

20. 일반적으로 류점수연산의 비정확성을 말할 때에는 거의 같은 량의 덜기에서 소거(cancellation)로 인하여 생기는 오유를 념두에 둔다. 그러나 X 와 Y 가 근사적으로 같을 때에는 차 X-Y 가 오유가 없이 정확히 얻어 진다. 이 말은 실지로 무엇을 의미하는가?

21. 콤퓨터에서 리용한 임의의 류점수표현은 어떤 실수만을 정확히 표현할수 있다. 즉 실수를 제외한 다른 모든 수들은 근사적으로 표현된다. A′가 실제값 A 의 근사값이라고 하면 그때 상대오차 r 는 다음과 같이 표현된다.

$$r = \frac{A - A'}{A}$$

    다음의 10 진수 ＋ 0.4 를 밑수=2, 지수부는 4bit(편위방식)이고 결수부는 7bit 인 류점수형식으로 표현하시오. 그리고 상대오차를 구하시오.,

22. 수값 A 와 B 가 근사값 A′, B′로 콤퓨터에 기억된다. 임의의 잘라버리기 혹은 둥

그리기오유를 무시하고 이 두수의 적의 상대오차가 근사적으로 개별적인 수들에서의 상대오차들의 합으로 된다는것을 증명하시오.

23. A = 1.427일 때 잘라 버리는 방법으로 A를 1.42로 둥그리기한 경우와 A를 1.43으로 둥그리기한 경우의 상대오차를 각각 구하시오.

24. 콤퓨터계산에서 생기는 가장 심각한 오유들중의 하나는 두개의 서로 비슷한 수들이 덜어 질 때 생긴다. A = 0.22288과 B = 0.22211인 경우 이 문제를 고찰하시오. 콤퓨터는 소수점아래 다섯번째 자리부터 잘라 버린다고 가정한다.

결국 A′= 0.2228이고 B′= 0.2221이다.

ㄱ. A′와 B′에 대한 상대오차는 얼마인가?

ㄴ. C′= A′- B′에 대한 상대오차는 얼마인가?

25. 다음과 같은 류점수더하기가 어떻게 수행되는가를 보여 주시오(여기서 결수부는 소수점아래 4자리까지로 본다.).

ㄱ. $0.5566\times10^{3}+0.7777\times10^{3}$

ㄴ. $0.3344\times10^{2}-0.8877\times10^{-1}$

26. 다음과 같은 류점수덜기가 어떻게 수행되는가를 보여 주시오(여기서 결수부는 소수점아래 4자리까지로 본다.).

ㄱ. $0.7744\times10^{-2}-0.6666\times10^{-2}$

ㄴ. $0.8844\times10^{-2}-0.2233\times10^{0}$

27. 다음의 류점수계산이 어떻게 수행되는가를 보여 주시오(여기서 결수부는 소수점아래 4자리까지로 본다.).

ㄱ. $0.2255\times10^{2}\times0.1234\times10^{1}$

ㄴ. $0.8833\times10^{3}\div0.5555\times10^{5}$

28. 다음의 8진수를 16진수로 표현하시오.

ㄱ- 12, ㄴ- 5655, ㄷ- 2550276, ㄹ- 7654336, ㅁ- 3726755

29. 최종적으로 2진표현을 가지는 모든 실수(2진소수점의 오른쪽에 유한개의 수자들을 가짐)들은 역시 최종적인 10진표현(10진소수점의 오른쪽에 유한개의 수자들을 가짐)을 가진다는것을 증명하시오.

부록 8. 수체계

1. 10진수체계

일상생활에서는 수를 표현하기 위하여 10진수(0,1,2,3,4,5,6,7,8,9)에 기초한 수체계를 리용하고 있는데 바로 이 수체계를 10진수체계라고 한다. 이제 10진수 83이 무엇을 의미하는가를 보기로 하자. 83은 8개의 10과 3을 더한것을 의미한다.

$$83 = (8\times10)+3$$

수 4728은 4개의 1000,7개의 100,2개의 10 그리고 8을 모두 더한것을 의미한다.

$$4728 = (4 \times 1000) + (7 \times 100) + (2 \times 10) + 8$$

10진수체계는 밑수 10을 가진다고 말할수 있다. 이것은 10진수에서 매 수자가 그 수자의 자리에 따라 지수가 붙은 10을 곱한다는것을 의미한다. 례를 들면 다음과 같다.

$$83 = (8 \times 10^1) + (3 \times 10^0)$$
$$4728 = (4 \times 10^3) + (7 \times 10^2) + (2 \times 10^1) + (8 \times 10^0)$$

소수값은 다음과 같은 형식으로 표현한다.

$$472.83 = (4 \times 10^2) + (7 \times 10^1) + (2 \times 10^0) + (8 \times 10^{-1}) + (3 \times 10^{-2})$$

일반적으로 $X = \{\ldots x_2 x_1 x_0 \ldots x_{-1} x_{-2} x_{-3} \ldots\}$의 10진수표현에서 X의 값은 다음과 같이 표현된다.

$$X = \sum_i x_i 10^i$$

2. 2진수체계

10진수체계에서는 10개의 서로 다른 수자들이 밑수 10으로 수들을 표현하는데 리용된다. 2진수체계는 2개의 수자 1과 0만을 가진다. 따라서 2진수체계에서 수들은 밑수 2로 표현된다.

혼돈이 없이 정확히 구별하기 위하여 때로는 그 수의 밑수를 첨자로 써주기도 한다. 실례로 83_{10}과 4728_{10}은 10진수표기로 표현된 수 혹은 간단히 10진수들이다. 2진수표기에서 수자 1과 0은 10진수표기에서와 같은 의미를 가진다 .

$$0_2 = 0_{10}$$
$$1_2 = 1_{10}$$

10 진수표기에서와 같이 보다 큰 수를 표현하기 위하여 2 진수의 매 수자는 자리에 따라 서로 다른 값을 가진다.

$$10_2 = (1 \times 2^1) + (0 \times 2^0) = 2_{10}$$
$$11_2 = (1 \times 2^1) + (1 \times 2^0) = 3_{10}$$
$$100_2 = (1 \times 2^2) + (0 \times 2^1) + (0 \times 2^0) = 4_{10}$$

소수값인 경우에는 소수점아래값이 부의 제곱으로 표현된다.

$$1001.101 = 2^3 + 2^0 + 2^{-1} + 2^{-3} = 9.625_{10}$$

3. 2진수와 10진수사이 변환

2진수를 10진수로 바꾸는것은 간단하다. 이미 앞에서 이와 관련한 여러가지 실례들

을 보았다. 2진수를 10진수로 바꾸자면 매 2진수자리에 알맞는 제곱의 2을 곱하고 그것들을 모두 더하면 된다.

10진수를 2진수로 바꿀 때에는 옹근수부와 소수부를 따로따로 취급해야 한다. 이제 10진수옹근수 N을 2진수로 바꾸는 경우를 고찰해 보자. 10진수 N을 2로 나누어 상 N_1과 나머지 R_1이 얻어 진다고 하면 다음과 같이 쓸수 있다.

$$N = 2\times N_1 + R_1 \qquad R_1=0 \text{ 혹은 } 1$$

다음에 상 N_1을 또 2로 나눈다. 이때 새로운 상과 나머지를 각각 N_2, R_2이라고 하면 다음과 같이 표현된다.

$$N_1 = 2\times N_2 + R_2 \qquad R_2=0 \text{ 혹은 } 1$$

결국

$$N=2(2N_2+R_2)+R_1=2^2N_2+R_2\times 2^1\times R_1\times 2^0$$

마찬가지로 상 N_2이 $N_2 = 2N_3 + R_3$으로 된다면 N은 다음과 같이 표현된다.

$$N=2^3N_3+R_3\times 2^2+R_2\times 2^1\times R_1\times 2^0$$

$N>N_1>N_2$이므로 이 과정를 반복하면 결과적으로 상 $N_k=1$과 0혹은 1로 되는 나머지 R_k가 나올것이다(10진옹근수 0, 1이 2진옹근수 0, 1과 등가이므로 이것은 제외시킨다.).

$$N=(1\times 2^k)+(R^k\times 2^{k-1})+\cdots+(R_3\times 2^2)+(R_2\times 2^1)+(R_1\times 2^0)$$

결국 2에 의한 나누기를 반복함으로써 밑수가 10인 수를 밑수가 2인 수로 바꾼다. 이렇게 하여 상이 0이 될 때까지 나머지는 무게가 증가하는 순서로 10진옹근수 N의 2진수가 표현된다. 이상의 실례들을 그림 8-27에 보여 주었다

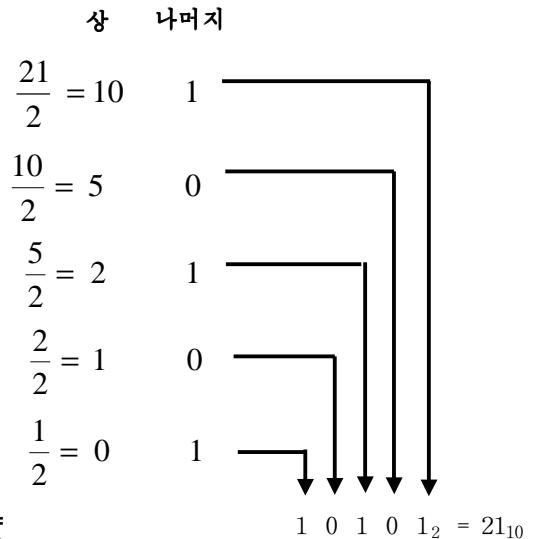
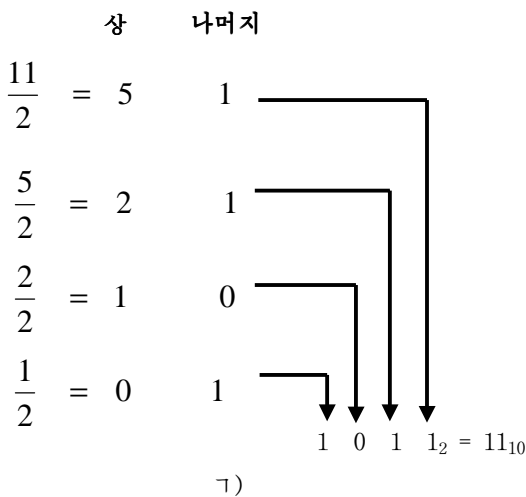



그림 8-27. 10진옹근수를 2진수로 변환하는 실례 ㄱ- 11_{10}, ㄴ- 21_{10}

다음으로 소수부는 그림 8-28 에서 보여 준바와 같이 2 에 의한 반복곱하기로 얻어진다. 매 단계에서 10 진수의 소수부에는 2 가 곱해 진다. 2 의 곱하기결과로 생긴 적에서 10 진소수점의 왼쪽에 있는 수자는 0 혹은 1 이며 이것이 2 진수를 표현한다. 이 표현은 가장 무게가 큰 비트로부터 출발한다. 적의 소수부는 다음단계에서 곱해지는수로 리용된다. 이것이 정확한가를 보기 위하여 이제 정의 10 진소수점 F<1 을 생각하자. F 는 다음과 같이 표현할수 있다.

$$F=\left(a_{-1}\times\frac{1}{2}\right)+\left(a_{-2}\times\frac{1}{2^2}\right)+\left(a_{-3}\times\frac{1}{2^3}\right)+\cdots$$

여기서 매 a_{-i} 는 0 혹은 1 이다. 웃식의 량변에 2 를 곱하면 다음과 같이 된다.

$$2F=a_{-1}+\left(a_{-2}\times\frac{1}{2}\right)+\left(a_{-3}\times\frac{1}{2^2}\right)+\left(a_{-4}\times\frac{1}{2^3}\right)+\cdots$$

두 표현식에서 옹근수부는 같아야 한다. $2F$ 의 옹근수부는 0<F<1이므로 0혹은 1이여야 하며 결국 이 값은 간단히 a_{-1}이다. 따라서 $2F$ 를 다음과 같이 쓸수 있다.

$$F_1=\left(a_{-2}\times\frac{1}{2}\right)+\left(a_{-3}\times\frac{1}{2^2}\right)+\left(a_{-4}\times\frac{1}{2^3}\right)+\cdots$$

a_{-2} 을 구하기 위하여 이 처리를 반복해 보자. 그러면 이 처리가 정확치 못하다는것을 알수 있다. 자리수가 유한인 10 진소수가 자리수가 무한인 2 진소수가 될수 있다. 이와 같은 경우에 보통 변환알고리듬은 요구하는 정확성에 따라 이미 규정된 수의 단계를 벗어 나면 변환을 중지한다.

그림 8-28. 소수로 된 10 진표기를 2 진표기로 바꾸는 실례
ㄱ-$0.81_{10}=0.110011_2$ (근사값), ㄴ-$0.25_{10}=0.01_2$ (정확한 값)

4. 16 진수표기

수자형콤퓨터구성요소들은 고유한 2 진속성을 가지므로 콤퓨터들에서 모든 형의 자료는 여러가지 2 진부호로 표현된다. 본문에서의 2 진부호와 옹근수의 2 진표기에 대한 실례들은 이미 앞에서 보았다. 앞으로 다른 형의 자료에 2 진부호를 리용한 실례들을 더 고찰하겠다. 2 진수체계는 콤퓨터에서는 매우 쓰기 편리하지만 인간에게는 매우 시끄러운것으로 되고 있다. 그리하여 콤퓨터로 일하는 대다수 콤퓨터 전문가들은 보다 함축된 표기를 리용하고 있다.

그들이 과연 어떤 표기를 리용하는가? 그중의 하나가 바로 10진표기이다. 이 표기는 2 진표기보다 어느 정도 함축된 표기이지만 밑수 2 와 밑수 10 사이의 변환이 오랜것으로 하여 쓰기 불편하다.

그리하여 10진표기대신 16진표기를 쓴다. 16진표기에서는 우선 2진수자들이 4개씩 묶어 진다. 이렇게 묶어 진 4개의 2진수자들의 가능한 조합에 다음과 같이 기호를 붙인다.

0000 = 0	1000 = 8
0001 = 1	1001 = 9
0010 = 2	1010 = A
0011 = 3	1011 = B
0100 = 4	1100 = C
0101 = 5	1101 = D
0110 = 6	1110 = E
0111 = 7	1111 = F

16 개의 기호가 리용되므로 이 표기를 16 진표기라고 하며 16 개의 기호를 16 진수자라고 한다.

16 진수자렬은 밑수 16 을 리용하여 10 진옹근수로 표현할수 있다. 례를 들면 다음과 같다.

$$1A_{16} = (1_{16} \times 16^{1}) + (A_{16} \times 16^{0}) =$$
$$= (1_{10} \times 16^{1}) + (A_{10} \times 16^{0}) = 26$$

16진표기는 밑수 16을 가진 옹근수를 표현하는데만 리용되는것이 아니다. 이 표기는 본문, 수 그리고 다른 형의 자료표현에도 리용하며 또 이러한 16진표기들에서 임의의 2진수자렬을 간결하게 표현하는데도 리용된다. 16진표기를 쓰는 리유는 다음과 같다.

1. 2진표기보다 더 함축된다.
2. 대다수 콤퓨터에서 2 진자료는 4bit 의 배수로 되며 따라서 단일 16 진수자의 배수로 된다.
3. 2진표기와 16진표기사이 변환이 매우 쉽다.

2진표기와 16진표기사이 변환과 관련한 실례로써 2진수렬 110111100001을 고찰하면 다음과 같다.

1101	1110	0001	$= DE1_{16}$
D	E	1	

숙련된 프로그람작성자인 경우 종이장우에 써보지 않고도 2진자료를 등가적인 16진수로 변환할수 있다. 이를 위한 숙련은 필요 없지만 독자들이 16진표기와 부닥칠수도 있으므로 이 론의를 본문에서 취급한다.

큰끝주소배치

바이트 주소								
00	11 00	12 01	13 02	14 03	04	05	06	07
08	21 08	22 09	23 0A	24 0B	25 0C	26 0D	27 0E	28 0F
10	31 10	32 11	33 12	34 13	'A' 14	'B' 15	'C' 16	'D' 17
18	'E' 18	'F' 19	'G' 1A	1B	51 1C	52 1D	1E	1F
20	61 20	62 21	63 22	64 23				

◆◆

- ◆ 콤퓨터명령의 본질적요소에는 수행되는 연산을 규정하는 조작코드, 연산의 입력 및 출력을 지정하는 원천 및 목적연산수참조, 늘 암시적인 다음명령참조 등이 있다.
- ◆ 조작코드는 일반적인 연산종류들가운데서 어느 한 연산을 규정한다. 연산종류에는 산수-론리연산, 두 등록기사이나 등록기와 기억기사이 혹은 두 기억기위치들사이 자료전송, 입력/출력과 조종 등이 있다.
- ◆ 연산수참조는 연산수자료의 등록기 혹은 기억기위치를 규정한다. 자료의 형은 주소, 수, 문자, 론리자료 등 이다.
- ◆ 처리장치들에서 공통적인 구조적특징은 탄창을 리용하는것이다. 탄창은 프로그람작성자가 볼수도 있고 보지 못할수도 있다. 탄창은 틀을 호출하거나 그로부터 돌아갈 때 리용하며 또 기억기주소지정에도 리용할수 있다. 기본탄창연산은 PUSH, POP 이며 이외에 꼭대기 하나 혹은 두개의 탄창위치들에 대한 연산도 있다. 탄창은 대체로 높은 주소에서 시작하여 낮은 주소로 가면서 채워 지도록 실현된다.
- ◆ 처리장치들은 기억기에 다중바이트(여러개의 바이트)들을 배치할 때 바이트의 배치순서에 따라 큰끝, 작은끝, 쌍끝처리장치로 분류할수 있다. 가장 낮은 수값주소에 맨 웃자리바이트가 기억된 여러 바이트수값은 큰끝방식으로 기억된다고 말한다. 가장 높은 수값주소에 맨 웃자리바이트가 기억되는 경우는 작은끝방식이라고 한다. 쌍끝처리장치는 두 방식을 다 지원할수 있는 처리장치이다.

◆◆

이 책에서 론의하는 대부분 내용들은 콤퓨터사용자나 프로그람작성자들에게는 명확지 않은 점들이 적지 않다. 프로그람작성자가 Pascal 이나 Ada 와 같은 고급언어를 리용하는 경우에는 콤퓨터의 근본을 이루는 기본방식의 매우 적은 부분만을 볼수 있다.

콤퓨터설계가나 콤퓨터프로그람작성자가 콤퓨터를 조사할수 있는 하나의 경계분야는 기계명령모임이다. 설계가의 립장에서 보면 콤퓨터기계명령모임은 CPU 의 기능적인 요구를 보장하여 준다. CPU 를 동작시키는것은 크게 보면 기계명령모임의 집행을 동반하는 과제이다. 사용자의 견지에서 보면 사용자가 기계어(실지로는 아셈불리언어, 제 6 절 참고)로 프로그람을 작성할수 있게 한다. 다시말하여 사용자가 등록기와 기억기구조, 콤퓨터가 직접 지원하는 자료의 형, ALU 의 기능 등을 알고 프로그람을 작성할수 있게 한다.

콤퓨터의 기계명령모임에 대한 서술에서는 콤퓨터의 CPU 를 설명하는데 많은 몫을 돌려 졌다. 이 장과 다음장에서는 먼저 기계명령에 기본중점을 두고 그다음에는 CPU 의 구조와 기능을 고찰한다.

제 1 절. 기계명령의 특성

CPU 의 동작(연산)은 그것이 실행하는 **기계명령** 혹은 **콤퓨터명령**이라고 하는 명령들에 의하여 결정된다. CPU 가 실행할수 있는 각이한 명령들의 모임을 CPU **명령모임**이라고 한다.

1. 기계명령의 요소

모든 명령은 그것이 CPU 에 의하여 실행되므로 CPU 에서 요구하는 정보를 포함하고 있어야 한다. 그림 3-6 을 되풀이한 그림 9-1 은 명령실행에서의 단계를 명확히 보여주고 있으며 또 기계명령의 요소들을 정의하고 있다.

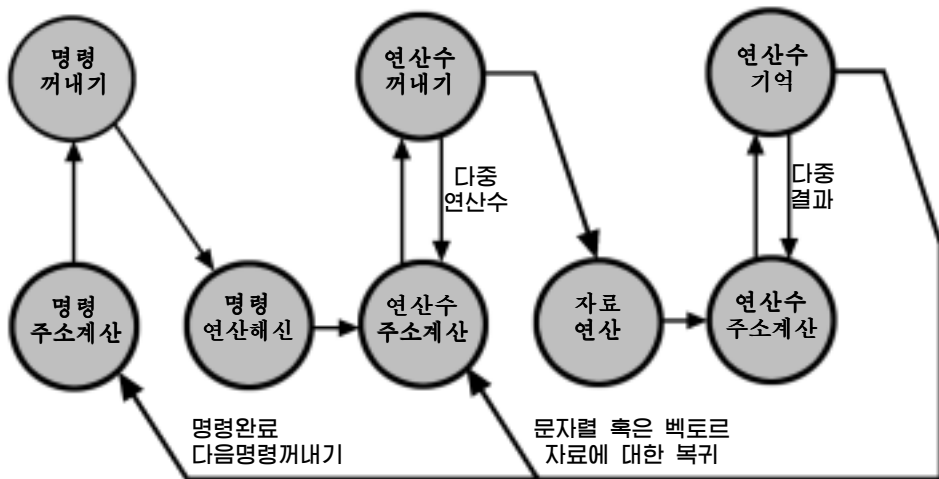



그림 9-1. 명령주기상태도

기계명령의 요소들은 다음과 같다.

- **조작코드:** 수행되는 연산을 규정한다. 더하기, 입력/출력과 같은 연산종류를 규정한다. 연산은 조작코드라고 하는 2 진부호로 규정된다.
- **원시연산수참조:** 연산에는 하나 혹은 그이상의 원천연산수들이 참가한다. 이 연

산수는 연산에서 입구로 되는 연산수이다.
- **결과연산수참조:** 연산은 결과를 낳는다.
- **다음명령참조:** 이것은 명령실행이 완료된후에 CPU 가 다음명령을 불러 내도록 한다.

불러 들인 다음명령은 주기억기에 들어 가며 가상기억기체계인 경우에는 주기억기 혹은 외부기억기(디스크)에 들어 간다. 대부분 경우에 불러 들인 다음명령은 현재명령이 집행된 다음 즉시 실행된다. 명시적인 참조를 하자면 주기억기 혹은 가상기억기주소가 지정되여야 한다. 주소가 지정되는 형식은 제 10 장에서 고찰한다.

원천 및 결과연산수들은 다음의 세가지 령역중 어느 하나에 있을수 있다.

- **주기억기 혹은 가상기억기:** 다음명령참조에서와 같이 주기억기 및 가상기억기의 주소가 지정되여야 한다.
- **CPU 등록기:** 특별한 경우를 제외하고 CPU 는 기계명령에 의해 참조되는 하나 혹은 그이상의 등록기들을 가지고 있다. 한개의 등록기만 있는 경우 그에 대한 참조는 암시적이다. 등록기가 하나이상 있는 경우에는 매 등록기에 유일번호가 할당되며 따라서 명령은 자기가 요구하는 등록기의 번호를 가지고 있어야 한다.
- **입출력장치:** 명령은 연산을 위하여 입력/출력(I/O)모듈이나 장치를 규정할수 있어야 한다. 기억기배치식 I/O 를 리용하는 경우 이것은 주기억기 혹은 가상기억기의 주소로 된다.

2. 명령표현

콤퓨터에서 모든 명령은 비트렬로 표현된다. 명령은 그 명령의 성분요소들에 대응하는 마당으로 분할된다. 명령형식의 간단한 실례를 그림 9-2 에 보여 주었다. 다른 하나의 실례인 IAS 명령형식은 이미 그림 2-2 에서 보았다. 보통 대다수 명령모임에서는 한가지 이상의 형식이 리용된다. 명령은 실행되는 동안 CPU 에 있는 명령등록기(IR)에 보존된다. CPU 는 이 등록기에 있는 여러 마당으로부터 자료를 꺼내여 필요한 연산을 수행할 수 있다.

그림 9-2. 간단한 명령형식

이 책을 읽는 프로그람작성자들이나 독자들은 기계명령의 2 진표현을 다루기가 힘들수 있다. 그리하여 기계명령의 기호표현을 리용하는것이 공통적인 추세, 실천으로 되였다. 그 실례가 바로 표 2-1 에서 고찰한 IAS 명령모임이다.

일반적으로 조작코드는 **명령략어**(mnemonics)로 표현되며 이것은 연산을 가리킨다. 실례를 몇가지 들면 다음과 같다.

ADD	더하기
SUB	덜기
MPY	곱하기

DIV　　　　나누기
LOAD　　　기억기로부터 자료를 넣기
STOR　　　기억기에 자료를 기억

연산수 역시 기호로 표현된다. 실례로 명령 ADD R, Y는 등록기 R의 내용에 자료위치 Y에 있는 값을 더한다는것을 의미한다. 실례에서 Y는 기억기위치를 보여 주는 주소이며 R는 특별한 등록기에 해당한다. 이 연산에서는 기억기위치의 내용을 더할뿐 그 주소에 대한 연산을 하지 않는다.

이로부터 기계어프로그람을 기호양식으로 표현할수 있다. 모든 기호조작코드는 고정된 2진표현을 가지며 프로그람작성자는 모든 기호연산수의 위치를 규정한다. 실례로 프로그람작성자는 다음과 같은 수값정의로부터 프로그람을 작성한다.

X = 513
Y = 514

이 단순한 프로그람은 기호입구를 접수하여 그것을 조작코드 및 2진형태의 연산수참조로 바꾸고 2진기계명령을 만든다.

현재 기계어프로그람작성자들은 그리 많지 못하다. 오늘날 대부분 프로그람들은 고급언어 혹은 아쎔블리언어로 작성되는데 여기서 아쎔블리언어는 이 장의 마감부분에서 론의한다. 그러나 기호기계어는 기계명령을 서술하기 위한 쓸모 있는 도구로 남아 있으며 이러한 목적을 위해서 앞으로도 계속 리용될것이라고 본다.

3. 명령의 종류

BASIC나 FORTRAN과 같은 고급언어명령을 하나 고찰해 보자.

X = X + Y

이 명령문은 Y에 기억된 값을 X에 기억된 값에 더하여 그 결과를 X에 넣으라고 콤퓨터에 지시한다. 이것이 기계명령에서는 어떻게 실현되겠는가? 이것을 보기 위하여 위치 513과 514에 대응하는 변수 X와 Y를 생각하자. 단순한 기계명령모임이 있다고 하면 이 연산은 세가지 단순명령으로 수행할수 있다.

1. 기억기위치 513의 내용을 등록기에 넣는다.
2. 그 등록기에 기억기위치 514의 내용을 더한다.
3. 기억기위치 513에 등록기의 내용을 기억한다.

이 실례로부터 알수 있는것은 하나의 BASIC 명령이 3개의 기계명령을 필요로 할수 있다는것이다. 이것이 바로 고급언어와 기계어사이의 대표적인 관계이다. 고급언어는 변수를 리용하여 간결한 대수양식으로 연산을 표현한다. 기계어는 등록기에로 혹은 등록기로부터의 자료전송을 동반하는 기본적인 양식으로 연산을 표현한다.

이 단순한 실례를 가지고 이제 실지 콤퓨터에 반드시 포함되여야 할 명령의 형들을 고찰해 보자. 콤퓨터는 사용자가 임의의 자료처리과제를 수행할수 있도록 명령모임을 가지고 있어야 한다. 이것을 고찰하기 위한 한가지 방도는 고급프로그람작성언어의 능력을 고찰하는것이다. 고급언어로 작성한 임의의 프로그람은 그것이 실행되자면 기계어로 번역되여야 한다. 따라서 기계어명령모임은 고급언어로부터 임의의 명령을 표현할수 있도록 충분히 많아야 한다. 이와 같은 견지에서 기계어명령형들을 다음과 같이 나누어 고찰할수 있다.

- **자료처리:** 산수-론리연산명령
- **자료기억:** 기억명령
- **자료전송:** I/O 명령
- **조종:** 검사 및 분기명령

산수연산명령은 수값자료를 처리하는 계산능력을 제공한다. **론리**명령은 수가 아니라 단어의 비트들에 대한 연산을 한다. 이 명령은 사용자가 쓰고 싶어하는 임의의 다른 형의 자료를 처리할수 있는 가능성을 준다. 이 연산은 기본적으로 CPU 등록기에 있는 자료에 대하여 수행된다. 그러므로 기억기와 등록기사이 자료를 이동하기 위한 기억명령이 있어야 한다. **I/O** 명령은 기억기에 프로그람이나 자료를 넣어 계산결과를 사용자에게 보여 주기 위하여 필요한 명령이다.**검사** 혹은 **시험**명령은 자료단어의 값을 검사하거나 혹은 계산상태를 검사하는데 리용되는 명령이다. **분기**명령은 판정방식에 따라 서로 다른 명령모임으로 갈라 지는데 리용하는 명령이다.

앞으로 여러가지 형태의 명령들을 보다 구체적으로 고찰한다.

4. 주소의 수

처리장치의 기본방식을 론하는 전통적인 방법들중 한가지 방법은 모든 명령에 포함되여 있는 주소의 개수에 대하여 론하는것이다. 이 론의는 CPU 설계가 점점 복잡해 짐에 따라 큰 의의를 가지지는 못하지만 기본방식의 차이를 표현하고 분석한다는 견지에서는 지금도 효과적인 방법으로 되고 있다.

그러면 명령하나에 필요한 주소의 수는 최대로 얼마로 되겠는가? 명령들가운데서 산수론리연산명령들은 명백히 가장 많은 연산수들을 요구한다. 실지로 모든 산수론리연산명령들은 하나 혹은 두개의 연산수들을 가지고 있다. 이로부터 연산수들을 참조하는데 두개의 주소가 최대값으로 된다. 연산결과는 제3의 주소를 정의하고 거기에 기억해야 한다. 명령실행이 끝나면 다음명령이 불리워 지므로 그의 주소가 또 요구된다.

명령	해설
SUB Y, A, D	Y ← A−B
MPY T, D, E	T ← D×E
ADD T, T, C	T ← T+C
DIV Y, Y, T	Y ← Y÷T

ㄱ)

명령	해설
MOVE Y, A	Y ← A
SUB Y, B	Y ← A−B
MOVE T, D	T ← D
MPY T, E	T ← T×E
ADD T, C	T ← T+C
DIV Y, T	Y ← Y÷T

ㄴ)

명령	해설
LOAD D	AC ← D
MPY E	AC ← AC×E
ADD C	AC ← AC+C
STOR Y	Y ← AC
LOAD A	AC ← A
SUB B	AC ← AC−B
DIV Y	AC ← AC÷Y
STOR Y	Y ← AC

ㄷ)

그림 9-3. Y=(A−B)÷(C+D×E) 집행프로그람
ㄱ- 3주소명령, ㄴ- 2주소명령, ㄷ- 1주소명령

이러한 고찰로부터 하나의 명령은 4개의 주소참조를 포함할수 있다고 볼수 있다. 즉 2개의 연산수, 하나의 결과, 다음명령의 주소 등이 포함될수 있다. 실지로는 4개의 주소를 가진 명령이 매우 드물다. 대다수 명령들을 다음명령의 주소가 확정되여 있으므로 하나, 둘 혹은 세개의 연산수주소만을 가진다. 여기서 다음명령주소는 프로그람계수기로부터 얻어 지게 된다.

그림 9-3 에서는 Y = (A − B) ÷ (C + D × E)를 계산하는데 리용할수 있는 대표적인 하나, 둘, 세개의 주소를 가진 명령들을 비교하였다. 세개의 주소를 가지면 매 명령이 2 개의 연산수위치와 결과위치를 지정할수 있다. 임의의 연산수위치값을 변화시키지 않는 경우 림시기억위치 T 가 도중의 결과를 기억하는데 리용된다. 결국 5 개의 연산수를 리용하여 4 개의 명령으로 이것을 수행할수 있다.

세개의 주소를 가진 명령형식은 세가지 주소참조를 위하여 상대적으로 긴 명령형식을 요구하므로 일반적으로 리용하기는 불편하다. 두개의 주소를 가진 명령에서는 2 진연산을 위하여 두 주소중 하나의 주소가 연산수 및 결과기억용으로 리용되여야 한다. 따라서 명령 SUB Y, B는 계산 Y-B를 수행하고 Y에 그 결과를 기억한다. 두개의 주소를 가진 명령형식은 필요한 기억공간을 줄이지만 일부 불편한 점도 있다. 즉 연산수의 값이 바뀌지 않도록 하기 위하여 MOVE 명령을 리용하는데 이 명령은 연산이 수행되기전에 결과 혹은 림시기억위치로 연산수값들중 하나를 옮긴다. 그림 9-3 에서 이 프로그람은 두개의 주소를 가진 6 개의 명령을 리용하여 작성되였다.

제일 간단한것은 하나의 주소만을 가진 명령이다. 이 명령이 정확히 수행되도록 하자면 두번째 주소는 이미 확정되여 있어야 한다. 이 명령은 지난 시기의 콤퓨터들에서도 공통적으로 리용되였다. 두번째 주소의 확정은 CPU 등록기인 **축적기** 혹은 AC 를 리용하여 진행한다. 축적기는 어느 하나의 연산수를 가지며 연산결과를 기억하는데 리용된다. 하나의 주소만을 가진 명령을 리용하면 Y 계산을 위한 산수연산은 그림 9-3 의 ㄷ에서 알수 있는바와 같이 8 개의 명령으로 수행할수 있다.

사실상 일부 명령에 대하여서는 0 주소로 할수 있다. 0 주소명령은 탄창이라고 하는 특별한 기억기조작에 적용할수 있다. 탄창의 조작은 후입선출 즉 마지막에 들어 간것이 제일 먼저 나오는 규칙에 준하고 있다. 탄창은 정해 진 위치에 있으며 적어도 꼭대기 두개의 요소들은 CPU 등록기내에 있다. 그러므로 0 주소명령은 두개의 탄창요소를 참조한다. 탄창에 대하여서는 부록 9-1 에서 구체적으로 론의하였다. 탄창의 리용에 대하여서는 이 장과 제 10 장에서 후에 더 론의한다.

표 9-1 은 령, 하나, 둘 그리고 세개의 주소를 가진 명령에 대한 설명을 개괄하여 보여 주고 있다. 표에서는 모든 경우에 대하여 다음명령의 주소가 확정되여 있으며 하나의 연산이 두개의 원천연산수와 한개의 결과연산수를 가지고 수행된다고 가정하였다.

표 9-1. 명령주소의 응용 (비분기명령)

주소의 개수	기호표현	설명
3	OP A, B, C	A ← B OP C
2	OP A, B	A ← A OP B
1	OP A	AC ← AC OP A
0	OP	T ← T OP (T-1)

AC =축적기
T =탄창의 꼭대기
A, B, C =기억기 혹은 등록기위치

명령당 주소의 수는 설계를 위하여 결정해야 할 기초적인 수이다. 명령당 주소의 수를 적게 하면 보다 원시적인 명령들로 되여 CPU 의 복잡성을 줄이며 또한 길이가 보다

짧은 명령으로 된다. 따라서 일반적으로 실행시간이 보다 길어 지며 프로그람이 복잡해진다. 1 주소명령과 여러 주소명령사이에는 등록기리용에서 중요한 한계가 있다. 하나의 주소명령을 리용하는 경우 프로그람작성자는 일반적으로 하나의 일반목적등록기인 축적기만을 리용할수 있다. 여러 주소명령을 쓰는 경우에는 공통적으로 여러개의 일반목적등록기들을 리용할수 있다. 등록기들을 여러개 리용하게 되면 일부 연산을 등록기로만 할수 있다. 보통 등록기참조는 기억기참조보다 속도가 빠르므로 이렇게 하면 프로그람의 집행속도를 높일수 있다. 이와 같은 여러개 등록기리용의 유연성과 효과성으로 하여 대부분 현대콤퓨터들에서는 둘 혹은 세개의 주소를 가진 명령을 쓴다.

명령당 주소의 수를 얼마로 선정하겠는가 하는것은 여러가지 인자들의 영향을 고려하여 선정해야 할 복잡한 문제이다. 문제는 주소가 기억기위치를 참조하는가 아니면 등록기를 참조하는가 하는데 있다. 등록기의 개수가 작다면 보다 작은 비트들이 등록기참조에 요구된다. 또한 다음장에서 고찰하면 알수 있겠지만 콤퓨터는 여러가지 형태의 주소지정방식들을 제공하며 이 방식의 규정에는 하나 혹은 그 이상의 비트들을 리용한다. 결론적으로 대다수 CPU 설계에서는 여러가지 명령형식의 설계가 동반된다.

5. 명령모임설계

콤퓨터설계측면에서 볼 때 가장 흥미가 있을뿐아니라 이미 대부분이 해석되여 있는것중의 하나는 명령모임설계이다. 명령모임설계는 그것이 콤퓨터체계의 많은 부분들에 영향을 미치므로 매우 복잡하다. 명령은 CPU 에서 수행하는 많은 기능들을 정의하며 따라서 CPU 의 실현에서 매우 큰 의의를 가진다. 명령모임은 CPU 를 조종하는 프로그람작성자의 뜻이다. 결국 프로그람작성자의 요구는 명령모임설계에서 고찰해야 할 문제로 된다.

명령모임설계와 관련한 일부 가장 근본적인 고찰이 아직 미해결로 남아 있다고 하면 놀랄 사람들이 적지 않을것이다. 오히려 최근년간에 이 근본적인 고찰에서의 불일치성이 점점 더 커지고 있다. 가장 중요한 기본설계자료들은 대체로 다음과 같다.

- **연산지령표 혹은 연산저장고:** 제공되는 연산의 크기와 위치, 복잡성
- **자료형:** 연산에 참가하는 여러가지 형의 자료
- **명령형식:** 비트단위로 표현된 명령의 길이, 주소의 개수, 여러가지 마당의 크기 등
- **등록기:** 명령에 참조할수 있는 CPU 등록기의 수와 그것들의 리용
- **주소지정:** 연산수의 주소를 규정하는 방식 혹은 방식들

명령모임설계에서는 우의 설계자료들이 호상 련관되여 있으므로 모두 함께 고찰해야 한다. 물론 일정한 순서에 따라서 이것들을 고찰해야 하지만 그것들의 호상관계를 고찰하는것이 기본으로 된다.

이상과 같은 론의는 매우 중요하므로 제 3 편의 많은 내용들이 명령모임설계로 되여 있다. 이 절에 뒤이어 이 장에서는 자료의 형과 연산저장고(operation repertoire)에 대하여 고찰한다. 제 10 장에서는 주소지정방식과 명령형식을 론하며 제 12 장에서는 축소명령모임콤퓨터(RISC)를 보기로 한다. RISC 기본방식에서는 많은 현대콤퓨터들에서 리용한 명령모임설계사상을 대부분 의문시하고 있다. RISC 기계의 한 실례가 바로 PowerPC 이다. 이 장과 다음장에서 PowerPC 를 고찰한다. 그러나 RISC 의 관계에 기초한 PowerPC 의 의미는 제 12 장에서 론의한다.

제 2 절. 연산수의 종류

기계명령은 자료에 대한 연산을 진행한다. 자료의 종류에는 대체로 다음과 같은것이 있다.

- 주소
- 수
- 문자
- 론리자료

주소는 실제상 한가지 자료의 양식이다. 앞으로 제 10 장에서 주소지정방식을 론의하면 이것을 보다 명백히 알수 있다. 많은 경우에 일부 계산은 명령에서의 연산수참조로 주기억기 혹은 가상기억기주소를 결정하여 수행되여야 한다. 이런 상황에서는 주소를 부호 없는 옹근수로 고찰할수 있다.

이외에도 수,문자,론리자료 등의 자료형이 있으며 이것들도 이 절에서 간단히 고찰한다. 일부 콤퓨터들에서는 이와 같은 형식의 자료외에도 특별한 자료형이나 자료구조를 리용하고 있다. 실례로 문자목록 혹은 문자렬에 대하여 직접 연산을 진행하는 기계연산자도 있다.

1. 수

모든 기계어는 수값자료형을 가진다. 지어 비수값자료처리에서도 계수기, 마당폭 등으로 작용하는 수들이 요구될수 있다. 보통 수학에서 리용되는 수들과 콤퓨터에 기억된 수들사이의 중요한 구별은 수자에 제한성이 있다는것이다. 그 리유는 우선 콤퓨터에서 표현할수 있는 수의 크기가 제한된다는것이며 둘째로는 류점수인 경우에 그것들의 정확도에 제한성이 있다는것이다. 따라서 프로그람작성자는 둥그리기, 자리넘침, 아래자리넘침 등이 있는 결과를 얻게 된다.

다음의 세가지 형의 수값자료는 모든 콤퓨터들에서 공통적으로 리용되고 있다.

- 옹근수 혹은 고점수
- 류점수
- 10 진수

우의 수값자료에서 첫 두개의 수값자료는 제 8 장에서 이미 상세히 고찰하였다. 여기서는 10 진수에 대하여 좀더 고찰한다.

콤퓨터의 모든 내부연산이 본성적으로 2 진수로 진행된다고 해도 체계의 사용자인 사람은 10 진수를 리용한다. 그러므로 입력에서 10 진수를 2 진수로, 출력에서 2 진수를 10 진수로 바꾸어야 한다. I/O 를 많이 리용하는 응용이나 상대적으로 작고 또 상대적으로 간단한 계산에서는 10 진수형태로 수를 기억하거나 처리하는것이 더 좋다. 이를 위하여 조임형 10 진수를 가장 일반적인 표현으로 리용한다.

조임형 10 진수에서는 모든 10 진수자가 4bit 코드로 표현된다. 즉 0=0000, 1=0001, … , 8=1000, 9=1001 로 표현된다. 이것은 가능한 16 개의 4bit 값들중에서 10 개만을 리용하므로 비효과적인 코드로 된다. 수를 형성할 때에는 보통 8bit 의 배수로 되게 4bit 코드들이 서로 한줄에 서게 된다. 례를 들면 246 의 코드는 0000001001000110 으로 된다. 이 코드는 명백히 직접 2 진수표현보다는 덜 압축된 형식이지만 변화준비작업을 요구하지 않는다.

부의 수들은 조임형 10 진수자렬의 왼쪽 혹은 오른쪽에 4bit 부호자리를 추가하여 표현한다. 실례로 코드 1111 은 〈-〉부호를 의미한다.

대다수 콤퓨터들은 조임형 10 진수에 대한 연산을 직접 수행하는 산수명령을 가지고 있다. 알고리듬은 제 3 절에서 론의한 내용과 매우 류사하지만 반드시 10 진자리올림연산을 고려해야 한다.

2. 문 자

자료는 보통 본문이나 문자렬로 이루어 진다. 이러한 본문자료가 사람에게는 가장 편리하지만 그것을 문자그대로는 자료처리 및 통신체계에 쉽게 기억하거나 전송할수 없다. 이러한 체계들은 2 진자료를 대상으로 하여 설계된다. 그리하여 수많은 코드들이 문자를 비트렬로 표현할수 있도록 제안되였다. 가장 대표적인 실례는 모르스(Morse)코드이다. 현재 가장 일반적으로 리용되는 문자코드는 ASCII (American Standard Code for Information Interchange)로 규정된 IRA(International Reference Alphabet)코드이다(표 6-1). 이 코드에서 매 문자는 단일한 7bit 패턴으로 표현된다. 따라서 이 코드를 리용하면 128 개의 서로 다른 문자를 표현할수 있다. 이것은 인쇄할수 있는 총 문자수를 넘는 큰 수이므로 패턴들중 일부는 조종문자로 리용한다. 조종문자들중에서 일부는 한 페지씩 문자를 인쇄하는데 리용하며 또 일부는 통신순서조종에 리용되기도 한다. IRA 로 부호화된 문자들은 대체로 문자당 8bit 로 기억되거나 전송된다. 여기서 8 번째 비트는 0 으로 설정되거나 오유검출용기우성비트로 리용되기도 한다. 기우성비트로 리용되는 경우 이 비트는 매 바이트에서 2 진수 1 의 총수가 늘 기수(기수기우성)이거나 늘 우수(우수기우성)가 되도록 설정된다.

표 6-1 에서는 IRA 비트렬 011XXXX 에 대하여 수자 0 ~ 9 를 그것들의 오른쪽 4bit 가 0000 ~ 1001 인 등가적인 2 진수로 표현하였다. 이것은 조임형 10 진수와 같은 코드로써 7bit IRA 와 4bit 조임형 10 진수표현사이의 변환을 쉽게 해준다.

문자를 부호화하는데 리용되는 다른 하나의 코드는 확장된 2 진식 10 진상호변환코드(EBCDIC)이다. EBCDIC 는 IBM S/370 콤퓨터에서 리용되고 있다. 이 코드는 8bit 코드이다. IRA 와 마찬가지로 EBCDIC 는 조임형 10 진수와 호환성이 있다. EBCDIC 에서는 코드 11110000 ~ 11111001 이 수자 0 ~ 9 를 표현한다.

3. 론리자료

보통 모든 단어 혹은 다른 주소지정단위(바이트, 반단어 등)들은 단일한 자료단위로 취급된다. 그러나 매 비트가 0 혹은 1 의 값을 가지는 n 개의 1bit 요소로 자료를 구성한 n 비트단위를 리용하는것이 때로는 더 효과적이다. 자료를 이런 방법으로 고찰하는 경우 이 자료를 **론리자료**라고 한다.

자료를 비트단위로 고찰하면 두가지 우점이 있다. 첫째로, 때때로 매 요소들이 오직 값 1(참)과 값 0(거짓)만을 가질수 있는 불 혹은 2 진자료요소들의 배렬을 기억해야 할 필요성도 제기될수 있다. 이때 론리자료로 처리하면 기억기를 가장 효과적으로 리용할수 있다. 둘째로, 자료요소의 비트들을 조작하려는 경우에 그 우점이 나타난다. 실례로 류점수연산을 프로그람적으로 수행한다면 일부 연산들에서는 무게를 가진 비트들을 밀기해야 할 필요성이 제기된다. 다른 하나의 실례로서 ASCII 코드를 조임형 10 진수로 바꾸자면 매 바이트의 오른쪽 4bit 를 추출해야 한다.

이상의 실례를 통하여 같은 자료가 때로는 론리자료로 취급되기도 하고 때로는 수값

혹은 본문자료로도 된다는것을 알수 있다. 자료단위의 《형》은 그 자료를 대상하는 연산에 의하여 결정된다. 이것은 고급언어(실례로 Pascal)에서는 일반성을 가지지 않지만 기계어에서는 거의나 일반적인 경우로 된다.

제3절. Pentium Ⅱ와 PowerPC의 자료형식

1. Pentium Ⅱ의 자료형

Pentium Ⅱ는 길이가 8(바이트), 16(단어), 32(배단어), 64bit(4배단어)인 자료형을 처리한다. 자료의 구조와 효과적인 기억기응용의 유연성을 최대로 높이기 위하여 단어들은 우수주소에 배치하지 않는다. 마찬가지로 배단어들은 4로 나누어 지는 주소에 배치하지 않으며 4배단어들은 8로 나누어 지는 주소에 배치하지 않는다. 그러나 32bit 모선을 통하여 자료가 호출되는 경우에는 4로 나누어 지는 주소에서 시작하여 배단어단위로 진행된다. 처리장치는 잘못 배치된 값들에 대한 요구를 모선전달을 위한 요구렬로 바꾼다. Pentium Ⅱ는 모든 Intel 80x86계렬과 마찬가지로 작은끝격식(little-endian style)을 리용한다. 즉 맨 아래자리바이트가 제일 낮은 주소에 기억된다(끝배치(endianness)에 대한 고찰은 부록 9-2 참고).

표 9-2. Pentium Ⅱ의 자료형

자 료 형	설명
일반형	임의의 2진수를 내용으로 가지는 바이트, 단어(16bit), 배단어(32bit), 4배단어(64bit)위치들
옹근수형	2의 보수표현을 리용하는 바이트,단어 혹은 배단어로 된 부호 없는 2진수값
순서자료형	바이트, 단어 혹은 배단어로 된 부호 없는 옹근수
비조임 2진식 10진수 (BCD)형	매 바이트로 하나의 수자를 표현하는 0～9범위에 있는 BCD수자의 표현
조임 BCD형	두개의 BCD수자들의 조인 바이트표현, 0～99범위의 값을 가진다.
가까운 지시기형	한 토막내에서의 변위를 표현하는 32bit의 유효주소. 비토막화된 기억기에서는 모든 지시기에 대하여, 토막화된 기억기에서는 한토막에서의 참조를 위해 리용한다.
비트마당형	매 비트의 자리가 독립적인 단위로 고찰되는 련속비트렬은 임의의 바이트의 임의의 비트에서 시작할수 있으며 2^{32}-1개의 바이트를 가질수 있다.
바이트렬형	0～2^{32}-1개의 바이트를 가지는 바이트, 단어 혹은 배단어로 된 련속바이트렬
류동소수점수형	그림 9-4 참고.

바이트, 단어, 배단어와 4배단어는 일반자료형이다. 추가적으로 Pentium Ⅱ는 개별적인 명령에 의하여 인식되고 연산되는 특수한 여러가지 자료형들의 배렬을 제공하고 있다. 표 9-2는 이러한 형의 자료들을 개괄하여 보여 주고 있다.

류점수형은 실제적으로 류동소수점수장치에서 리용하며 류점수명령에 의해 그 연산이 진행되는 자료형의 한 종류이다. 그림 9-4에서 보여 준바와 같이 류점수명령은 류동소수점수는 물론 옹근수와 조임형10진옹근수에 대한 연산도 할수 있다. 옹근수들은 보통

2 의 보수로 표현된 수이며 16, 32, 64bit 의 긴 형이다. 조임형 10 진옹근수는 0~9 범위의 수를 18 개의 수자(한 수자는 4bit)를 가진 부호-크기표현으로 기억된다. 이 세가지 류점수표현는 IEEE 754 표준에 부합된다.

s = 부호비트(0 = 정수; 1 = 부수)
d_n = 10 진수자(바이트당 2)
X = 비트들이 의미가 없음; 넣기에서 무시, 기억에서 령
D = 암시적인 소수점의 자리
I = 결수부의 옹근수비트;림시 실수로 기억, 단정확도 및 배정확도에서는 암시적임

그림 9-4. Pentium II 수값자료의 형식

2. PowerPC 의 자료형식

PowerPC 는 길이가 8(바이트), 16(반단어), 32(단어), 64bit(배단어)인 자료형을 처리한다. 일부 명령에서는 기억기연산수들이 32bit 한계로 일치되여야 한다. 그러나 일반적으로는 이렇게 되지 않는다. PowerPC 의 한가지 흥미 있는 특징은 작은끌 혹은 큰끌격식을 모두 리용할수 있다는것이다. 즉 맨 아래자리바이트는 가장 높은 주소 혹은 가장 낮은 주소에 기억될수 있다(끌배치는 부록 9-2 참고).

바이트, 반단어, 단어와 배단어는 일반자료형이다. 처리장치는 명령에 따라 주어 진

자료요소의 내용을 해석한다. 고점수처리장치인 경우는 다음과 같은 자료형을 인식한다.

- **부호 없는 바이트형:** 론리연산 혹은 옹근수산수연산에 리용될수 있다. 전체 등록기크기만큼 왼쪽에 0을 확장하여 기억기에서 일반등록기에로 자료를 넣는다.
- **부호 없는 반단어형:** 부호 없는 바이트형과 같지만 자료의 크기가 16bit이다.
- **부호 있는 반단어형:** 산수연산에 리용된다. 전체 등록기크기만큼 왼쪽에 부호비트를 확장하여 기억기에 넣는다(모든 빈 자리에 부호비트가 들어 간다.).
- **부호 없는 단어형:** 론리연산에 리용되며 주소지시기로도 리용된다.
- **부호 있는 단어형:** 산수연산에 리용한다.
- **부호 없는 배단어형:** 주소지시기로 리용한다.
- **바이트렬형:** 길이가 0∼128byte

이외에 PowerPC는 IEEE 754에서 정의한 단정확도 및 배정확도류점수형의 자료도 처리할수 있다.

제 4 절. 연산의 종류

여러가지 조작코드들의 수는 콤퓨터마다 다르다. 그러나 모든 콤퓨터들에서 연산의 서로 같은 일반형태들을 찾아 볼수 있다. 대표적인 연산종류에는 다음과 같은것들이 있다.

- 자료전송
- 산수연산
- 론리연산
- 수변환
- I/O
- 체계조종
- 조종이행

표 9-3([HAYE88]에 기초)에 매 연산종류에서 가장 많이 쓰이는 명령들을 목록으로 주었다. 이 절에서는 CPU가 표 9-4에서 보여 준 알맞는 형의 명령을 실행하는 동작을 간단히 론하면서 여러가지 형의 명령들을 간단히 개괄한다. 아래의 제목들은 제 11장에서 더 상세하게 검토될것이다.

표 9-3. 일반적인 명령모임연산

형	연산이름	설명
자료	이동(전송)	단어 혹은 불로크를 원천으로부터 목적으로 전송한다.
	기억	처리장치로부터 기억기에로 단어를 전송한다.
	넣기(꺼내기)	기억기로부터 처리장치로 단어를 전송한다.
	교환	원천 및 목적연산수의 내용을 교체한다.
	지우기(재설정)	목적연산수에 값이 0인 단어를 전송한다.
	설정	목적연산수에 값이 1인 단어를 전송한다.
	밀어넣기	원천연산수로부터 탄창의 꼭대기에로 단어를 전송한다.
	밀어꺼내기	탄창의 꼭대기에서 목적연산수로 단어를 전송한다.

표 계속

산수 연산	더하기	두 연산수의 합을 계산
	덜기	두 연산수의 차를 계산
	곱하기	두 연산수의 적을 계산
	나누기	두 연산수의 상을 계산
	절대값	연산수를 2의 절대값으로 재배치
	부정	연산수의 부호를 변화시킨다.
	증가	연산수에 1을 더한다.
	감소	연산수에서 1을 던다.
론리 연산	론리곱하기 론리더하기 론리부정(보수) 안맞음론리더하기	비트단위의 지정된 론리연산을 수행한다.
	검사	지정된 조건을 검사하여 그 결과에 따라 기발을 설정한다.
	비교	둘 혹은 그이상 연산수들은 론리적으로, 산수적으로 비교하여 그 결과에 따라 기발을 설정한다.
	조종변수설정	새치기조종, 시계조종 등의 보호목적을 위한 조종을 설정하는 명령부류
	밀기	끝에 상수를 도입하여 연산수를 왼쪽(오른쪽)으로 민다.
	회전	연산수의 두 끝을 결정한 상태에서 왼쪽(오른쪽)밀기를 한다.
조종 이행	이행(분기)	무조건적인 전송 즉 지정된 주소를 PC에 넣는다.
	조건이행	지정된 조건을 검사하여 그 조건에 따라 규정된 주소를 PC에 넣거나 아무것도 하지 않는다.
	보조루틴으로 이행	알려 진 위치에 현재프로그람조종정보를 놓는다. 지정된 주소로 이행한다.
	복귀	알려 진 위치로부터 PC와 다른 등록기내용을 재배치한다.
	실행	지정된 위치로부터 연산수를 불러 내여 명령으로서 실행한다. PC는 변경시키지 않는다.
	건너뛰기	PC를 다음명령을 건너 뛰도록 증가시킨다.
	조건건너뛰기	지정된 조건을 검사하여 조건에 따라 건너 뛰거나 뛰지 않는다.
	중지	프로그람집행을 중지한다.
	대기(유지)	프로그람집행을 멈추고 지정된 조건을 반복검사하여 조건이 반복되는 경우에 실행을 다시 시작한다.
	비연산	연산이 수행되지 않지만 프로그람실행은 계속된다.
입력/ 출력	입력(읽기)	지정된 I/O포구 혹은 장치로부터 목적연산수(주기억기 혹은 처리장치내의 등록기)에 자료를 전송
	출력(쓰기)	지정된 원천연산수로부터 I/O포구 혹은 장치에 자료를 전송
	I/O시작	I/O연산을 초기화하는 I/O처리장치에 명령을 전송
	I/O검사	I/O체계로부터 지정된 목적연산수에로 상태정보를 전송
변환	번역	대응표에 기초하여 기억기의 한 부분에 있는 값들을 번역한다.
	변환	한 형태에서 다른 형태에로 단어의 내용을 변환한다(조임형 10진수로 변환한다).

표 9-4. 각이한 형의 연산에서 CPU 의 동작

자료전송	한 위치에서 다른 위치에로 자료를 전송한다. 기억기를 리용하는 경우에는; 기억기주소를 결정한다. 가상기억기주소를 실지기억기주소로 변환한다. 캐쉬를 검사한다. 기억기읽기/쓰기를 초기화한다.
산수연산	산수연산의 전과 후에 자료전송이 동반된다. ALU 에서 기능을 수행한다. 조건코드와 기발을 설정한다.
론리연산	산수연산과 같다.
변환	산수 및 론리연산과 같다. 변환을 수행하는 특별한 론리가 동반될수 있다.
조종이행	프로그람계수기를 갱신한다. 보조프로그람의 호출/복귀를 위하여 파라메터넘기기와 련결을 관리한다.
I/O	I/O 모듈에 지령을 발행한다. 기억배치형 I/O 인 경우 기억배치주소를 결정한다.

1. 자료전송

기계명령의 가장 기본적인 형은 자료전송명령이다. 자료전송명령에서는 다음과 같은 것들을 규정해 주어야 한다. 첫째로, 원천 및 목적연산수의 위치를 규정해야 한다. 매 위치는 기억기나 등록기 혹은 탄창의 꼭대기일수 있다. 둘째로, 이동하는 자료의 길이가 지적되여야 한다. 셋째로, 연산수를 가진 모든 명령과 마찬가지로 자료전송명령에서도 매 연산수의 주소지정방식이 규정되여야 한다. 주소지정방식에 대하여서는 제 10 장에서 고찰한다.

명령모임에 들어 있는 자료전송명령의 선택은 설계가들이 어떤 합리적인 방안을 리용하였는가를 보여 준다. 실례로 연산수의 위치(기억기 혹은 등록기)는 일반적으로 조작코드나 연산수의 형식화에 의하여 지정할수 있다. 표 9-5 는 가장 일반적인 IBM S/370 의 자료전송명령을 보여 주고 있다. 자료전송명령에는 전송되는 자료의 량(8, 16, 32 혹은 64bit)을 지정하는 변량들이 있다.

표 9-5. IBM S/370 지료전송연산의 실례

연산략어	이 름	전송되는 비트수	설 명
L	넣기	32	기억기에서 등록기에로 전송한다.
LH	반단어넣기	16	기억기에서 등록기에로 전송한다.
LR	넣기	32	등록기에서 기억기에로 전송한다.
LER	넣기(짧은 형식)	32	류점수등록기에서 류점수등록기에로 전송한다.
LE	넣기(짧은 형식)	32	기억기에서 류동소수점수등록기에로 전송한다.
LDR	넣기(긴 형식)	64	류점수등록기에서 류점수등록기에로 전송한다.
ST	기억	32	등록기에서 기억기에로 전송한다.
STH	반단어기억	16	등록기에서 기억기에로 전송한다.
STC	문자기억	8	등록기에서 기억기에로 전송한다.
STE	기억(짧은 형식)	32	류점수등록기에서 기억기에로 전송한다.
STD	기억(긴 형식)	64	류점수등록기에서 기억기에로 전송한다.

또한 등록기에서 등록기로, 등록기에서 기억기에로, 기억기에서 등록기에로 자료를 전송하는 여러가지 명령들도 있다. VAX 콤퓨터와 비교해 보면 VAX 는 전송하는 자료의 각이한 크기를 표현하는 변량을 가진 이동(MOV)명령을 가지고 있다. 이 명령은 연산수가 등록기인가 혹은 연산수의 한 부분으로서의 기억기인가 하는것까지도 규정한다. 따라서 VAX방법은 프로그람작성자가 보다 적은 명령을 가지고 프로그람을 작성할수 있게 하므로 어느 정도 더 쉬워 보인다. 그러나 이 방법은 매 연산수의 위치(등록기 대 기억기)를 명령과는 독립적으로 규정해야 하므로 IBM S/370 방법보다는 어느 정도 명령이 더 긴 방법이다. 이 차이는 다음장에서 명령형식을 론하면서 고찰한다.

CPU 동작의 견지에서 볼 때 자료전송명령은 가장 단순한 명령형태이다. 원천 및 목적연산수들이 둘 다 등록기인 경우 CPU 는 간단히 자료가 한 등록기에서 다른 등록기에로 이동하게 한다. 이것은 CPU 내부에서의 연산이다. 하나 혹은 두개의 연산수들이 기억기에 있는 경우 CPU 는 다음과 같은 동작의 일부 혹은 모두를 수행해야 한다.

1. 주소지정방식에 기초하여 기억기주소를 계산한다(이 내용은 제 10 장에서 고찰).
2. 주소가 가상기억기와 관련되는 경우 가상기억기주소를 실기억기주소로 바꾼다.
3. 주소가 캐쉬주소인가를 판정한다.
4. 아니면 기억기모듈에 지령을 발행한다.

2. 산수연산

대다수 콤퓨터들은 더하기, 덜기, 곱하기, 나누기의 기본산수연산을 수행한다. 산수연산에서는 부호 있는 옹근수(고점수), 류점수, 조임형 10 진수 등을 리용한다.

산수연산명령에는 이외에도 여러가지 단일연산수명령들이 있다. 실례를 들면 다음과 같다.

- **절대값:** 연산수의 절대값을 취한다.
- **부수:** 연산수의 부호를 반전시킨다.
- **증가:** 연산수를 1 만큼 증가시킨다.
- **감소:** 연산수를 1 만큼 감소시킨다.

산수연산명령의 실행에는 ALU 에 입구하기 위한 연산수를 정하며 ALU 에서 계산된 결과를 그 출력에서 넘겨 받는 자료전송명령이 동반되게 된다. 그림 3-5 는 이 과정 즉 자료전송과 산수연산이 동반되는 과정을 보여 준다. 물론 ALU 는 이외에도 CPU 의 한 부분으로서 필요한 연산을 수행한다.

3. 론리연산

콤퓨터들은 보통 단어 혹은 기타 자료단위들의 개별적인 비트들을 조작할수 있는 여러가지 명령들을 가지고 있다. 이 연산들은 불연산(부록 1 참고)에 기초하고 있다.

불 혹은 2진자료에 대하여 수행되는 기본론리연산명령의 일부를 표 9-6에 보여 주었다.

표 9-6. 기본론리연산

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P=Q
0	0	1	0	0	0	1
0	1	1	0	1	1	0
1	0	0	0	1	1	0
1	1	0	1	1	0	1

NOT 연산은 한 비트를 반전시키며 AND, OR, XOR 는 각각 론리곱하기, 론리더하기, 안맞음론리더하기연산으로서 두 연산수를 리용하는 가장 일반적인 론리연산기능이다. EQUAL 은 효과적인 2 진수검사연산이다.

이와 같은 론리연산들은 n비트의 론리자료단위들에 비트단위로 적용할수 있다. 이제 두 등록기가 다음과 같은 자료를 가지고 있다면

(R1) = 10100101
(R2) = 00001111

이때

(R1) AND (R2) = 00000101

로 된다. 여기서 표현법(X)은 위치 X 의 내용을 의미한다. AND 연산은 한 단어에서 일정한 비트들을 선택하고 나머지는 령으로 지우는데 리용할수 있다. 다른 실례로 두 등록기에 각각 다음의 자료가 있다고 하자.

(R1) = 10100101
(R2) = 11111111

이때

(R1) XOR (R2) = 01011010

으로 된다. 이로부터 모든 비트가 1 인 단어가 안맞음론리더하기연산에 참가하면 다른 단어의 모든 비트가 반전된 결과가 얻어 진다는것을 알수 있다(1 의 보수).

비트별 론리연산외에도 대다수 콤퓨터들은 여러가지 밀기 및 회전기능을 가지고 있다. 가장 기본적인 연산을 그림 9-5 에 보여 주었다. **론리밀기**는 단어의 비트들을 왼쪽 혹은 오른쪽으로 미는 연산이다. 한끝에서 밖으로 밀리운 비트는 잃어 버리며 다른쪽 끝에는 0 이 들어 가게 된다. 론리밀기는 우선 한단어내에서 마당을 분리하는데 쓸수 있다. 단어안으로 밀려 들어 온 0 은 다른쪽 끝에서 밀려 밖으로 나간 불필요한 정보와 교체된다.

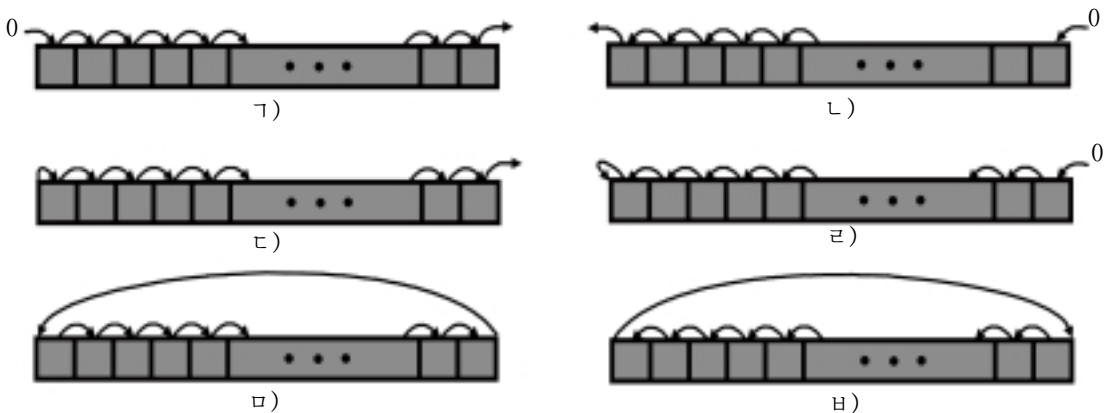



그림 9-5. 밀기 및 회전연산
ㄱ—론리오른쪽밀기, ㄴ—론리왼쪽밀기, ㄷ—산수오른쪽밀기,
ㄹ—산수왼쪽밀기, ㅁ—오른쪽 회전, ㅂ—왼쪽 회전

실례로 문자자료를 I/O 장치에 한번에 한 문자씩 전송하는 경우를 고찰해 보자. 매 기억기단어는 길이가 16bit 이므로 두 문자를 가질수 있으며 따라서 이 문자들을 보내기 전

에 한 문자씩 갈라야 한다. 한 단어에 있는 두 문자를 보내기 위하여 다음과 같이 한다.

1. 단어를 등록기에 넣는다.
2. 1111111100000000 과 등록기내용을 론리곱하기(AND)한다. 이것은 오른쪽에 있는 문자를 마스크하기 위한것이다.
3. 결과를 8번 오른쪽으로 민다. 이것은 등록기의 오른쪽 절반위치에 마스크되지 않은 다른 문자를 밀어 넣기 위한것이다.
4. I/O 를 수행한다. I/O 모듈은 자료모선에서 낮은 자리 8bit 를 읽는다.

이 단계는 한 단어에서 왼쪽문자를 전송하는 과정이다. 오른쪽문자를 보낼 때에는 다음과 같이 한다.

1. 다시 등록기에 단어를 넣는다.
2. 0000000011111111 과 등록기내용을 론리곱하기한다.
3. I/O 를 수행한다.

산수밀기연산은 부호 있는 옹근수로서 자료를 취급하며 부호비트는 밀리지만 내용은 변하지 않는다. 오른쪽 산수밀기에서 부호비트는 그의 오른쪽 비트위치로 옮겨 지지만 부호비트의 내용은 변하지 않는다. 이 연산들은 산수연산의 속도를 일정하게 높일수 있게 한다. 2 의 보수표기로 표현된 수들에 대한 왼쪽 및 오른쪽밀기는 자리넘침이나 아래자리넘침이 없는 경우 특별히 2 에 의한 곱하기 및 나누기에 대응한다.

회전 혹은 순환밀기연산은 연산에 참가하고 있는 모든 비트들을 보존한다. 회전연산의 한가지 특징은 매 비트를 련속적으로 제일 왼쪽 비트위치로 가져 가는것이다. 이것은 수로서 취급되는 자료의 부호를 검사하기 위한것으로 볼수 있다.

산수연산과 마찬가지로 론리연산에도 ALU 의 작용과 자료전송명령이 동반된다.

4. 변환

변환명령은 수의 형식을 변화시키거나 자료의 형식에 대한 연산을 진행하는 명령이다. 대표적인 실례는 10 진수를 2 진수로 바꾸는것이다. 보다 복잡한 편집명령의 실례는 S/370 변환(TR)명령이다. 이 명령은 하나의 8bit 코드를 다른 8bit 코드로 변환하는데 리용할수 있으며 이때 이 명령은 세개의 연산수를 가진다.

TR R_1, R_2, L

이 명령에서 연산수 R_2 는 8bit 코드표의 시작주소를 가진다. R_1 에서 지정된 주소에서 시작하는 L 개의 바이트들은 변환되여 매 바이트에 첨수가 붙은 표기입내용들로 재배치된다. 실례로 EBCDIC 를 ASCII 코드로 변환하기 위하여 우선 1000∼10FF 까지의 16 진수기억기위치에 256byte 의 표를 만든다. 이 표는 2 진수표현로 된 EBCDIC 코드의 순서로 ASCII 코드의 문자들을 가진다. 즉 ASCII 코드는 표에서 그 문자의 EBCDIC 코드의 2 진수값과 같은 상대적위치에 배치된다. 따라서 위치 10F0∼10F9 는 값 30∼39 를 가지게 된다. 이것은 F0 이 수자 0 에 대한 EBCDIC 코드이고 30 은 수자 0 에 대한 ASCII 코드이며 이런 방법으로 수자 9 까지 나가기때문이다. 이제 위치 2100 에 있는 수 1984 에 대한 EBCDIC 코드를 ASCII 코드로 변환하는 경우를 보자. 이를 위하여 다음과 같이 가정한다.

- 위치 2100∼2103 은 F1 F9 F8 F4 를 가진다.
- R_1 은 2100 을 가지고 있다.

• R_2는 1000을 가지고 있다.

이와 같이 가정하고 TR R_1, R_2, 4를 실행하면 위치 2100～2103에 31 39 38 34를 넣을 수 있다.

5. 입출력

입출력명령은 그 일부를 제 6장에서 상세히 고찰하였다. 여기서 본바와 같이 I/O에는 분리된 프로그람식 I/O, 기억기배치프로그람식 I/O, 직접기억기호출(DMA) 등을 비롯하여 여러가지 I/O가 있다. 또한 I/O처리장치의 리용도 여러가지이다. 입출력은 몇개의 I/O명령만으로 수행되는데 이 I/O명령들은 파라메터, 코드 혹은 지령단어에 의하여 규정되는 특별한 작용을 한다.

6. 체계조종

체계조종명령은 처리장치가 확실한 특권상태에 있거나 기억기의 특수한 특권령역안에서 프로그람을 집행하고 있는 동안에만 실행될수 있는 명령이다. 이 명령은 특별히 조작체계를 리용하기 위해 예약된것이다.

몇가지 체계조종연산의 실례를 들면 다음과 같다. 체계조종명령은 조종등록기를 읽거나 바꿀수 있다. 조종등록기에 대해서는 제 11장에서 고찰한다. 체계조종명령에는 또한 S/370 기억기체계에서 리용된것과 같은 기억기보호열쇠를 읽거나 변경하는 명령이 있다. 이외에도 다중프로그람작성체계에서 조종블로크를 처리하기 위한 호출명령도 있다.

7. 조종이행

지금까지 론의한 모든 형태의 연산들에서 다음에 실행할 명령은 기억기에서 현재명령의 직후에 배렬된 명령이다. 임의의 프로그람의 명령렬들가운데서 중요한 부분의 하나는 명령실행의 순서를 변화시키는 기능을 가지는것이다. 이 명령들에 대하여 CPU에 의하여 수행되는 연산은 프로그람계수기가 기억기에 있는 어떤 명령의 주소를 가지도록 갱신하는것이다.

조종이행연산이 필요한 리유에는 여러가지가 있다. 이 가운데서 가장 중요한것들은 다음과 같다.

1. 콤퓨터의 실지 리용에서는 모든 명령을 한번이상으로, 필요하다면 수천번이상 실행할수 있다. 또한 하나의 응용프로그람을 만들자고 해도 수천 혹은 수백만개의 명령들이 요구될수 있다. 이러한 경우에 개별적으로 모든 명령들을 일일이 쓴다는것은 생각할수도 없는 일이다. 만일 표자료라든가 항목들을 가진 목록자료라면 프로그람순환이 요구될수 있다. 이와 같은 경우에는 하나의 명령렬로 모든 자료를 처리할수 있도록 반복실행시킨다.
2. 실지 모든 프로그람들은 일부 판정조건을 만들어 리용한다. 콤퓨터는 한가지 조건이 주어 지면 그 일을 수행하고 다른 조건이 주어 지면 또 그 일을 수행할수 있어야 한다. 실례로 하나의 명령렬이 수의 두제곱뿌리를 계산하는 경우를 보자. 이때 이 명령렬의 시작부분에서는 우선 수의 부호가 검사된다. 수가 부수이면 계산을 수행하지 않고 오유조건이 보고된다.
3. 큰 프로그람 혹은 지어 중간크기정도의 프로그람이라고 해도 그것을 정확히 구

성하는것은 사실상 매우 어려운 과제이다. 한번에 하나의 프로그람으로 실행될수 있는 보다 작은 프로그람들로 과제를 분할하는 기구가 있다면 조종이행은 이것을 가능하게 한다.

아래에서는 가장 일반적인 조종이행연산명령들인 분기명령, 건너뛰기명령, 틀호출명령에 대하여 고찰한다.

분기명령

이행명령이라고 하는 분기명령은 실행해야 할 다음명령의 주소를 연산수로 가진다. 이러한 명령들은 보통 조건분기명령으로 된다. 분기는 일정한 조건이 만족되는 경우에만 이루어 진다. 다시말하여 프로그람계수기를 연산수에서 지정된 주소로 갱신한다. 만약 조건이 만족되지 않으면 다음명령이 실행된다(정상적으로 프로그람계수기를 증가시킨다.).

조건분기명령에서 조건을 발생시키는 일반적인 방법에는 두가지가 있다. 대부분 콤퓨터들은 일부 연산결과로 설정되는 한비트 혹은 여러 비트의 조건코드를 준다. 이 코드는 사용자가 볼수 있는 등록기라고 생각할수 있다. 실례로 산수연산(더하기, 덜기 등)은 다음과 같은 4가지 값 즉 0, +, -, 자리넘침가운데서 하나를 선정할수 있는 2bit 조건코드를 설정할수 있다. 이와 같은 콤퓨터에는 다음과 같은 4개의 서로 다른 조건분기명령들이 있다.

BRP X	결과가 정수인 경우 위치 X로 이행한다.
BRN X	결과가 부수인 경우 위치 X로 이행한다.
BRZ X	결과가 0이면 위치 X로 이행한다.
BRO X	자리넘침이 생긴 경우 위치 X로 이행한다.

우의 4가지 경우에 귀착되는 결과는 조건코드를 설정하는 가장 최근의 연산결과이다.

3개의 주소를 가진 명령형식에 리용할수 있는 다른 한가지 방법은 비교연산을 진행하여 그 명령에서 분기를 규정하는것이다. 실례를 들면 다음과 같다.

BRE R_1, R_2, X　　　R_1의 내용 = R_2의 내용이면 X로 이행

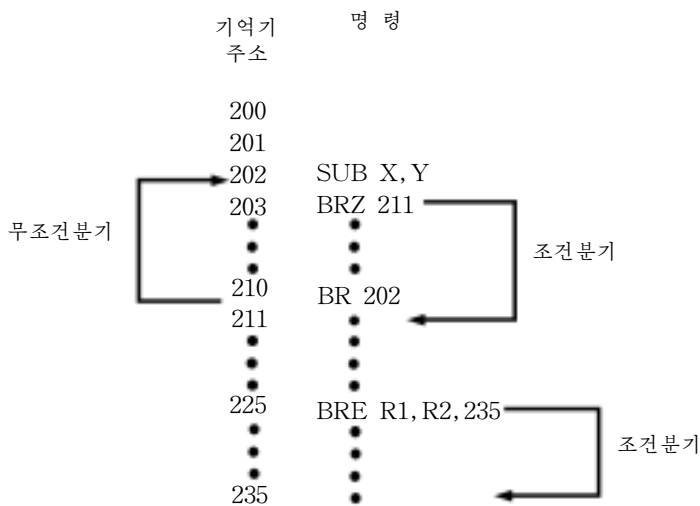



그림 9-6. 분기명령

그림 9-6에는 이러한 연산실례를 보여 주었다. 분기는 정방향(보다 높은 주소를 가진 명령) 혹은 역방향(보다 낮은 주소)으로 일어 날수 있다. 이 그림의 실례는 명령의 반복고리를 만드는데 무조건 및 조건분기가 어떻게 리용되는가를 보여 주고 있다. 즉 위치 202로부터 210 사이에 있는 명령들은 X 로부터 Y 를 던 결과가 0 으로 될 때까지 반복하여 실행된다.

건너뛰기명령

다른 한가지 조종이행연산명령은 건너뛰기명령이다. 건너뛰기명령은 암시적인 주소를 가지고 있다. 건너뛰기명령은 한개 명령을 뛰여 넘는다는것을 암시하므로 암시적인 주소는 다음명령의 주소와 한개 명령의 길이를 더한것으로 된다.

건너뛰기명령에는 목적주소마당이 필요 없으므로 다른 일은 하지 못한다. 대표적인 실례는 령인 경우 증가 및 건너뛰기(ISZ)명령이다. 이제 다음과 같은 프로그람의 토막을 생각하자.

```
301
•
•
•
309  ISZ R1
310  BR 301
311
```

이 토막에서는 반복순환을 만드는데 2개의 조종이행명령을 리용하고 있다. R_1에는 부의 반복수가 설정된다. 이 R_1은 순환의 끝에서 증가한다. R_1의 값이 0 이 아니면 프로그람은 고리의 시작으로 되돌아 온다. 만약 R_1의 값이 0이면 고리에서 벗어 나도록 건너뛰기 명령이 작용하며 고리의 끝다음명령에 의해 프로그람은 계속 실행된다.

틀호출명령

프로그람작성언어의 개발에서 가장 큰 혁신은 **틀**의 개념을 받아 들인것이다. 틀은 그자체가 하나의 프로그람으로서 보다 큰 프로그람에 들어 가는 프로그람이다. 틀은 프로그람안의 임의의 위치에서 **호출**할수 있다. 프로그람은 처리장치의 틀에 가서 그 틀을 실행하고 그다음 호출이 발생한 위치로 돌아 올것을 명령한다.

틀을 리용하는 리유는 경제성과 모듈화의 두가지이다. 우선 틀은 틀내의 명령렬들이 여러번 리용될수 있게 한다. 이것은 프로그람작성로력을 줄일수 있는 매우 경제적인 방법이며 또 체계내에서의 기억공간을 보다 효과적으로 리용할수 있게 한다. 틀은 또한 큰 프로그람작성과제를 보다 작은 분과제로 나눌수 있게 한다. 이것을 프로그람의 **모듈화**라고 하며 이것은 프로그람작성과제를 짧은 기간에 쉽게 수행할수 있게 한다.

틀기구에는 두개의 기본명령 즉 현 위치에서 틀에로 이행하는 호출명령과 틀로부터 그것이 호출된 위치에로 돌아 가는 복귀명령이 있다. 이 두 명령은 모두 분기명령에 속한다.

그림 9-7 ㄱ는 프로그람에서 리용되는 틀들을 보여 주고 있다. 이 실례에서는 주프로그람이 기억기위치 4000에 있다. 프로그람은 위치 4000에서 시작하면서 틀 PROC1을 호출한다. PROC1에 대한 호출명령을 만나게 되면 CPU는 주프로그람의 집행을 중지하고 위치 4500으로부터 다음명령을 불러 내여 PROC1의 실행을 시작한다. PROC1내에는 위치 4800에 있는 PROC2에 대한 두번의 호출이 있다. PROC2이 호출될 때마다 PROC1

실행은 중단되고 PROC2이 실행된다. RETURN명령은 호출한 프로그람으로 돌아 가서 CPU 가 CALL 명령다음의 명령실행을 계속하게 한다. 이와 같은 과정을 그림 9-7 ㄴ에 보여 주었다.

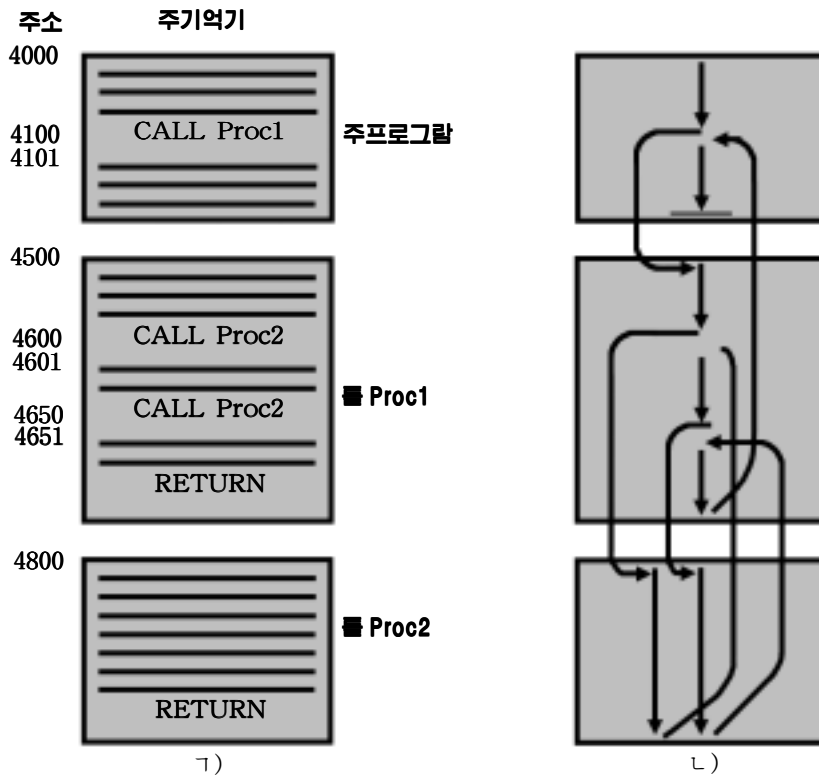



그림 9-7. 겹쌓인 틀: ㄱ-호출과 복귀, ㄴ-실행순서

쓸모 있는 틀에 대한 몇가지 결론들은 다음과 같다.

1. 틀은 여러 위치에서 호출할수 있다.
2. 임의의 틀안에서는 또 다른 틀을 호출할수 있다. 이것은 틀의 **겹놓임**을 임의의 길이로 할수 있게 한다.
3. 매틀에 대한 호출은 호출된 프로그람에서의 복귀로 끝난다.

임의의 위치에서 틀을 호출할수 있으므로 CPU 는 그에 맞게 복귀가 반드시 동반되도록 복귀주소를 보관해야 한다. 복귀주소 보관방법에는 다음의 세가지가 있다.

- 등록기
- 틀의 시작
- 탄창의 꼭대기

이제 기계어명령 CALL X를 생각하자. 이 명령은 위치 X에 있는 **틀에 대한 호출**이다. 등록기방법을 리용하는 경우에 CALL X는 다음과 같은 동작을 수행한다.

RN ← PC + △
PC ← X

여기서 RN은 늘 복귀주소보관목적으로 리용되는 등록기이다. PC는 프로그람계수기이고 A는 명령길이이다. 호출된 틀은 후에 복귀에 리용할 RN의 내용을 보관한다.

두번째 방법은 틀의 시작부에 복귀주소를 기억하는것이다. 이 경우에 CALL X 는 다음과 같은 동작을 수행한다.

$X \leftarrow PC + \triangle$
$PC \leftarrow X + 1$

이 동작은 매우 간단하며 복귀주소는 안전하게 보관된다.

이상의 두 방법들은 복귀주소보관에 효과적으로 리용되고 있다. 단지 이 방법들의 제한성은 **재진입**가능한 틀을 리용하지 못한다는것이다. 재진입가능한 틀은 같은 시간에 여러개의 호출이 그것을 열수 있는 틀이다. 순환틀(그자체를 호출하는 틀)이 바로 이 특징을 리용한 대표적인 실례로 된다.

보다 더 일반적이며 쓸모 있는 방법은 탄창을 리용하는것이다(탄창에 대한 론의는 부록 9-1 참고). CPU는 호출할 때 탄창에 복귀주소를 보관한다. 또한 틀로부터 복귀할 때에는 탄창에 보관한 이 주소를 리용한다. 그림 9-8에 이와 같은 탄창의 리용에 대하여 보여 주었다. 틀호출에서는 이와 같이 복귀주소를 제공하는것외에 파라메터를 넘겨 주어야 한다. 보통 이 파라메터들은 등록기를 통하여 틀에 넘겨 진다.

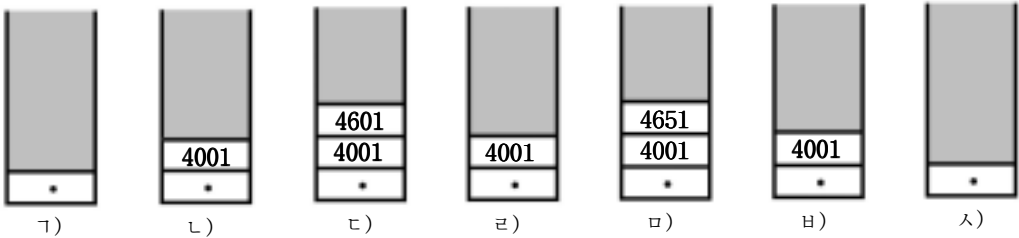



그림 9-8. 그림 9-7의 겹쌓인 보조루틴실현에서 탄창의 리용

ㄱ-초기탄창내용, ㄴ-CALL Proc1 후, ㄷ-초기 CALL Proc2, ㄹ-복귀후, ㅁ-CALL Proc2 후, ㅂ-복귀후, ㅅ-복귀후

이 방법외에도 CALL명령후 기억기에 파라메터들을 기억하는 방법이 있다. 이 경우에는 복귀주소를 파라메터가 보관되여 있는 다음위치에 기억해야 한다. 이 두가지 방법들은 부족점을 가지고 있다. 등록기를 리용하는 경우에는 등록기들이 알맞게 리용될수 있도록 호출된 프로그람과 호출하는 프로그람을 작성해야 한다. 기억기에 파라메터들을 기억하는 방법의 결함은 여러가지 값을 가질수 있는 파라메터들을 교환하기가 힘든것이다. 이 두 방법들도 재진입가능한 틀을 리용하지 못한다.

파라메터를 넘겨 주는 보다 유연한 방법은 탄창을 리용하는것이다. 이 방법에서는 처리장치가 틀을 호출할 때 복귀주소만 탄창에 보관하는것이 아니라 호출된 틀에 넘겨 주어야 할 파라메터까지도 탄창에 보관한다. 호출된 틀은 탄창으로부터 파라메터를 꺼내여 리용한다. 복귀할 때에도 복귀파라메터들이 탄창에 배치된다. 복귀주소를 비롯한 이와 같은 파라메터들의 전체 모임을 **탄창프레임**이라고 한다.

그림 9-9에는 틀호출과 관련한 탄창틀의 리용을 보여 주었다. 이 실례에서는 국부변수 X1과 X2가 선언되는 틀 P와 이 P에 의해 호출되는 국부변수 Y1과 Y2를 선언하는 틀 Q를 정의하고 있다. 그림에서 매틀의 복귀점은 대응하는 탄창들에 기억된 첫번째 항목이다. 그다음에는 앞틀의 시작을 가리키는 지시자를 기억한다. 이것은 탄창에 쌓아 지

는 파라메터들의 수나 길이가 변하는 경우에 필요하게 된다.

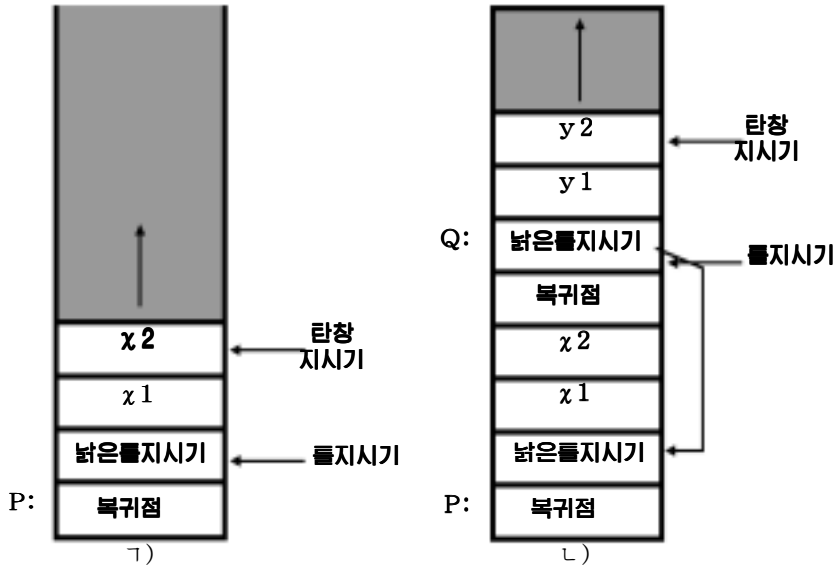



그림 9-9. 실례틀 P와 Q를 리용한 탄창틀의 증가[DEWA90]
ㄱ—P가 능동, ㄴ—P가 Q를 호출

제5절. Pentium II와 PowerPC의 연산종류

1. Pentium II의 연산종류

Pentium II는 여러개의 전문화된 명령을 비롯하여 복잡한 계렬의 연산형을 가지고 있다. Pentium II의 목적은 고급언어로 작성된 프로그람을 최적인 기계어로 번역하는 콤파일러의 작성도구를 제공하는것이였다. 표 9-7에 Pentium II의 연산종류와 그 실례들을 보여 주었다. 대부분 명령들은 일반적인 콤퓨터명령모임에서 찾아 볼수 있는 명령들이지만 일부는 80x86/ Pentium 기본방식에 알맞는것으로서 흥미가 있는것들도 있다.

표 9-7. Pentium II의 조작형태 (대표적인 연산실례를 포함)

명령	설명
	자료전송
MOV	등록기들사이 혹은 등록기와 기억기사이에서 연산수를 전송
PUSH	탄창에 연산수를 밀어넣기
PUSHA	탄창에 모든 등록기를 밀어넣기
MOVSX	바이트, 단어, 배단어, 확장된 부호를 전송. 바이트를 단어로, 단어를 배단어로 이동하며 이때 2의 보수부호확장을 한다.
LEA	유효주소 넣기. 목적연산수에 대한 값이 아니라 원천연산수의 변위를 넣는다.

표 계속

XLAT	표검사번역. 사용자코드번역표로부터 한바이트등록기인 AL 에 한바이트를 재배치한다. XLAT 가 실행될 때 AL 은 표에 대한 부호 없는 첨수를 가진다. XLAT 는 표첨수로부터 표기입까지 AL 의 내용을 변화시킨다.
IN, OUT	I/O 로부터 연산수를 입력, 출력
산수연산	
ADD	연산수들의 더하기
SUB	연산수들의 덜기
MUL	바이트, 단어 혹은 배단어연산수들과 단어, 배단어 혹은 4배단어의 결과를 가지는 부호 없는 옹근수곱하기
IDIV	부호 있는 나누기
론리연산	
AND	연산수들의 론리곱하기
BTS	비트검사와 설정. 비트마당의 연산수에 대한 연산을 한다 . 이 명령은 한비트의 현재값을 CF 기발에 복사하고 원래 비트를 1 로 설정한다.
BSF	정방향비트주사. 단어 혹은 배단어에서 1 인 비트를 주사하여 처음으로 만나는 1 의 번호를 등록기에 기억한다.
SHL/SHR	론리왼쪽 혹은 오른쪽밀기
SAL/SAR	산수왼쪽 혹은 오른쪽밀기
ROL/ROR	왼쪽 혹은 오른쪽회전
SETCC	상태기발에서 정의된 16 개의 조건에 따라 한바이트를 령 혹은 1 로 설정한다.
조종이행	
JMP	무조건이행
CALL	다른 위치로 조종을 넘기기. 조종을 이행하기전에 CALL 에 뒤따르는 배렬주소가 탄창에 보관된다.
JE/JZ	같거나 령인 경우 이행
LOOPE/LOOPZ	같거나 령인 경우 순환.이것은 등록기 ECX 에 기억된 값을 리용하는 조건이행이다. 명령은 분기조건을 위하여 ECX 를 검사하기전에 먼저 EXC 를 감소시킨다.
INT/INT0	자리넘침인 경우 새치기. 새치기봉사루틴에로 조종을 넘긴다.
문자렬연산	
MOVS	바이트, 단어, 배단어문자렬을 이동. 명령은 문자렬의 임의의 요소에 대하여 연산을 진행하는데 이 지적은 등록기 ESI 와 EDI 가 한다. 매 문자렬의 연산수등록기들은 자동적으로 문자렬의 다음요소를 지시하도록 증가 혹은 감소된다.
LODS	바이트, 단어, 배단어의 문자렬을 넣기
고급언어지원연산	
ENTER	블로크로 구조화된 고급언어의 규칙을 실현하는데 리용될수 있는 탄창틀을 창조
LEAVE	앞의 ENTER 의 동작과 반대인 연산
BOUND	배렬한계검사. 연산수1의 값은 두 한계값 즉 웃한계와 아래한계사이에 있다. 이 한계값들은 연산수2에서 참조하는 두 린접기억기위치내에 있다. 값이 이 한계범위를 벗어 나면 새치기가 일어 난다. 이 명령은 배렬첨수를 검사하는데 리용한다.
기발조종	
STC	자리올림기발설정
LAHF	A 등록기에 기발들을 넣기. A 등록기에는 SF, ZF, AF, PF 와 CF 비트들이 복사된다.
토막등록기	
LDS	지시기를 D 토막등록기에 넣기. 체계조종
HLT	정지

표 계속

LOCK	Pentium 이 LOCK 명령에 즉시 뒤따르는 명령이 있으면 이 명령의 리용을 배제하도록 공유기억기에 유지를 부탁
ESC	처리장치확장리탈. 다음 명령을 지적하는 리탈코드는 높은 정확도옹근수나 류점수계산을 진행하는 협동처리장치에 의해 실행된다.
WAIT	BUSY #가 취소될 때까지 대기. BUSY 단자는 협동처리장치가 실행을 끝냈다는것을 알려 주며 이 단자가 비활성(비능동)상태에 있다는것을 처리장치가 검출할 때까지 Pentium 프로그람실행을 중지한다.
보 호	
SGDT	전체 서술자표를 기억
LST	토막한계. 토막한계값은 사용자규정등록기에 넣는다.
VERR/VERM	읽기 및 쓰기토막을 확증
캐 쉬 관 리	
INVD	내부캐쉬를 지우기
WBINNVD	기억기에 버릴 행들을 써넣은후 내부캐쉬를 지우기
INVLPG	변환엿보기완충기(TLB)등록을 무효로 하기

호출 및 복귀명령

Pentium Ⅱ는 틀호출 및 복귀를 위한 4 가지 명령 즉 CALL, ENTER, LEAVE, RETURN 을 가지고 있다. Pentium Ⅱ에서는 그림 9-9 를 통하여 이미 고찰한 탄창틀을 리용하여 틀의 호출 및 복귀를 진행한다. 새로운 틀이 호출되는 경우 다음과 같은 동작이 진행된다.

- 복귀지점(복귀위치)을 탄창에 밀어 넣는다.
- 현재의 틀지시기를 탄창에 밀어 넣는다.
- 탄창지시기에 틀지시기의 새로운 값을 복사한다.
- 탄창지시기가 틀을 할당하도록 조정한다.

CALL 명령은 현재명령지시기값을 탄창에 밀어 넣고 명령지시기에 넣기점의 주소를 넣어 틀의 넣기점으로 이행하게 한다. 8088 과 8086 콤퓨터들에서 틀은 보통 다음의 명령렬들로 시작한다.

```
PUSH    EBP
MOV     EBP, ESP
SUB     ESP, space-for-locals
```

여기서 EBP 는 틀지시자이며 ESP 는 탄창지시기이다. 80286 과 그후의 콤퓨터들에서는 ENTER 명령이 우의 모든 연산을 단일한 명령으로 수행한다.

ENTER 명령은 콤파일러를 직접 지원할수 있게 만들어 진 명령모임에 속하는 명령이다. 이 명령은 Pascal, Cobol, Ada 와 같은 언어에서 겹쌓인 틀의 호출을 지원하는 특징도 가지고 있다(C 혹은 FORTRAN 에는 없다.). 이런 언어들에서는 겹쌓인 틀을 호출하는 보다 좋은 방법들이 있다. ENTER 명령은 PUSH, MOV, SUB 명령에 비하여 더 적은 바이트수의 기억기를 요구하지만(4byte 대 6byte) 실행시간이 더 길다(10 박자주기 대 6 박자주기). 결국 이 특징을 추가하는것이 명령모임설계가들에게는 좋은 구상을 주는것처럼 보이지만 이것은 처리장치의 실현을 복잡하게 만든다. 이 방법에 비해 볼 때 RISC 방법은 ENTER 와 같은 복잡한 명령을 피하면서도 보다 단순한 명령으로 효과적인 실현을 기대할수 있는 방법이다.

기억기관리

명령모임에는 기억기토막과 관련한 특별한 명령들도 있다. Pentium Ⅱ에는 오직 조작체계로만 실행될수 있는 특권명령들이 있다. 이 명령들은 국부적이며 전역적인 토막표(서술자표라고도 한다.)를 꺼내여 읽고 토막의 특권준위를 평가하기 위하여 그것을 검사하고 교체하는 역할을 한다.

한소편캐쉬와 관련한 특별한 명령들은 이미 제 4 장에서 고찰하였다.

조건코드

조건코드는 어떤 연산에 의하여 설정되여 조건분기명령에 리용되는 특별한 등록기내의 비트들이다. 이 조건들은 산수 및 비교연산에 의하여 설정된다. 대부분 언어들에서 비교연산은 덜기연산에서와 같이 두 연산수를 더는것이다. 덜기연산과의 차이점은 비교연산은 단지 조건코드만을 설정한다면 덜기연산은 목적연산수에 덜기결과를 기억한다는 것이다.

표 9-8 은 Pentium Ⅱ에서 리용되는 조건코드들을 목록화하여 보여 주고 있다. 매 조건 혹은 이 조건들의 결합은 조건이행에 리용한다. 표 9-9 는 조건이행조작코드를 정의한 조건들의 결합을 보여 주고 있다.

표 9-8. Pentium Ⅱ의 조건코드

상태비트	이름	설명
C	자리올림	산수연산에 이어 제일 왼쪽 비트자리에서 자리올림 혹은 자리내림이 있는가를 가리킨다. 그리고 일부 밀기 및 회전명령에 의해서 수정된다.
P	기우성	산수 및 론리연산결과의 기우성. 1 은 우수기우성, 0 은 기수기우성을 지시한다.
A	보조자리올림	AL 등록기를 리용하는 8bit 산수 혹은 론리연산에서 반바이트들사이 자리올림 혹은 자리내림을 표현한다.
Z	령	산수 및 론리연산결과가 0 이라는것을 지시한다.
S	부호	산수 및 론리연산결과의 부호를 지시한다.
O	자리넘침	더하기와 덜기연산후의 연산자리넘침을 지시한다.

이 목록에서 몇가지 흥미 있는 경우를 고찰해 보자. 우선 어느 수가 더 큰가를 결정하기 위하여 두 연산수들을 비교하는 경우를 보자. 이 문제는 수들이 부호가 있는가 없는가 하는데 달려 있다. 실례로 8bit 의 수 11111111 은 00000000 보다 더 크지만 그것은 어디까지나 이 두수가 부호 없는 옹근수(255>0)인 경우이다. 그러나 이 두수를 8bit 2 의 보수(-1<0)로 보면 11111111 이 00000000 보다 작다. 이리하여 많은 아쎔블리언어들은 이 두 경우를 구별하기 위하여 두가지 형태의 용어를 리용하고 있다. 즉 부호 있는 옹근수를 비교하는 경우에는 《보다 더 작다.》와 《보다 더 크다.》를 리용한다. 또한 부호 없는 옹근수로 보는 경우에는 용어《보다 아래》와 《보다 우》를 리용한다.

다음은 부호 있는 옹근수들의 비교에서 생기는 복잡성에 관한 문제이다. 부호 있는 결과는 다음과 같은 경우에 령보다 크거나 같게 된다. (1) 부호비트가 령이고 자리넘침이 없는 경우 (S=0 AND O=0), (2) 부호비트가 1 이고 자리넘침이 있는 경우이다. 이미 앞에서 고찰한 그림 8-5 는 여러가지 부호 있는 연산들에서 시험된 조건들이 정확히 맞는다는것을 보여 준다(문제 9-12 참고).

표 9-9. 조건이행와 SETcc 명령에서 PentiumⅡ의 조건

기호	검사조건	설명
A,NBE	C=0 과 Z=0	우 즉 아래가 아니거나 갈다(크다, 부호 없음).
AE,NB,NC	C=1	우이거나 갈다. 즉 아래가 아니다(크거나 갈다, 부호 없음).
B,NAE,C	C=1	아래 즉 우가 아니거나 갈다(작다, 부호 없음).
BE,NA	C=1 혹은 Z=1	아래이거나 갈다. 즉 우가 아니다(작거나 갈다. 부호 없음).
E,Z	Z=1	갈다. 즉 령 (부호 있음 혹은 부호 없음)
G,NLE	[(S=1 과 O=1) 혹은 (S=0 과 O=0)] 과 [Z=0]	크다. 즉 작지 않거나 갈다(부호 있음).
GE,NL	(S=1 과 O=0) 혹은 (S=0 과 O=0	크거나 갈다. 즉 작지 않다(부호 있음).
L,NGE	(S=1 과 O=0) 혹은 (S=0 과 O=1)	작다. 즉 크지 않거나 갈다(부호 없음).
LE,NG	(S=1 과 O=0) 혹은 (S=0 과O=1) 혹은 (Z=1)	작거나 갈다. 즉 크지 않다(부호 없음).
NE,NZ	Z=0	갈지 않다. 즉 령이 아니다(부호 없음 혹은 부호 있음).
NO	O=0	자리넘침이 없다.
NS	S=0	부호가 없다(부수가 아니다.).
NP,PO	P=0	기우성이 없다. 즉 기수기우성
O	O=1	자리넘침
P	P=1	기우성 즉 우수기우성
S	S=1	부호(부수이다.)

PentiumⅡ의 MMX 명령

1996년에 Intel은 자기의 Pentium계렬에 MMX기술을 받아 들였다. MMX는 다매체파제처리를 위하여 최적으로 만들어 진 명령들의 모임이다. MMX에는 단일명령, 다중자료(SIMD:Single-Instruction, Multiple-Data)방식으로 자료를 처리하는 57개의 새로운 명령들이 있다. 이 명령들은 더하기 혹은 곱하기와 같은 연산을 다중자료요소에 대하여 단번에 수행할수 있다. 매 명령은 단일박자주기로 명령을 실행한다. 이 고속병렬연산은 MMX명령을 리용하지 않는 알고리듬에 비해 연산속도가 2∼8배이다[ATKI96].

MMX 명령의 리용에서 기본은 다매체프로그람작성이다. 화상과 음성자료는 8 혹은 16bit와 같은 작은 자료형으로 된 큰 배렬로 이루어 지며 한편 일반적으로 명령은 32 혹은 64bit 자료에서 동작할수 있게 되여 있다. 도형처리기 및 화상에서는 하나의 장면이 화소 [1] 들의 배렬(모임)로 구성되며 매 화소 혹은 화소색성분(적, 록, 청)에 대하여 8bit가 할당된다. 또한 음성표본은 16bit 로 량자화된다. 일부 3D 도형그리기알고리듬에서는 기본자료형을 모두 32bit 로 하고 있다. 이러한 각이한 자료길이에서 병렬연산을 수행하기 위하여 MMX 에서는 세가지 새로운 자료형을 정의하고 있다. 매 자료형은 길이가 64bit 로서 여러개의 보다 작은 자료마당으로 이루어 졌으며 이 매 마당은 고정소수점옹근수를 가진다. 이 자료형들은 다음과 같다.

- **조입형바이트:** 64bit 크기로 묶어 진 8개의 바이트

[1] 픽셀 혹은 화소는 회색준위에 할당되는 수자화상의 가장 작은 요소이다. 등가적으로 화소는 그림의 점행렬표현에서 매 개별적인 점들에 해당한다.

- **조임형단어:** 64bit 로 묶어 진 4 개의 16bit 단어
- **조임형배단어:** 64bit 로 묶어 진 2 개의 32bit 배단어

표 9-10 은 MMX 명령모임을 목록화하여 보여 주고 있다. 대부분 명령들은 바이트, 단어 혹은 배단어로 병렬연산을 수행한다. 실례로 PSLLW 명령은 조임형단어연산수에 있는 4 개의 단어에 대하여 개별적으로 따로따로 왼쪽론리밀기를 수행한다. PADDB 명령은 조임형바이트연산수가 입력되고 조임형바이트가 출력되도록 독립적으로 매 바이트자리에 대하여 병렬더하기연산을 수행한다.

표 9-10. MMX 명령모임

종류	명령	설명
산수연산	PADD [B, W, D]	포장둥그리기를 가진 조임형 8byte, 4 개의 16bit 단어 혹은 2 개의 32bit 배단어의 병렬더하기
	PADD [B, W]	포화를 고려한 더하기
	PADD [B, W]	포화를 고려한 부호 없는 더하기
	PSUB [B, W, D]	포장둥그리기를 가진 덜기
	PSUBUS [B, W]	포화를 고려한 부호 없는 덜기
	PMULHW	32bit 결과에서 설정된 낮은 자리 16bit 를 가진 4 개의 부호 있는 16bit 단어의 병렬곱하기
	PMULLM	32bit 결과에서 선정된 낮은 자리 16bit 를 가진 4 개의 부호 있는 16bit 단어의 병렬곱하기
	PMADDWD	4 개의 부호 있는 16bit 단어의 병렬곱하기 즉 32bit 결과의 린접쌍들을 서로 더하기
비교	PCMPEQ [B, W, D]	같은가에 대한 병렬비교 즉 결과는 참(true)이면 1 들의 마스크, 거짓(false)이면 0 들의 마스크이다.
	PCMPGT [B, W, D]	더 큰가에 대한 병렬비교. 결과는 참이면 1 들의 마스크, 거짓이면 0 들의 마스크이다.
변환	PACKUSWB	단어를 부호 없는 포화연산을 하여 바이트로 묶기
	PACKSS [WB, DW]	부호 있는 포화연산을 하여 단어를 바이트로, 배단어를 단어로 묶기
	PUNPCKH [BW, WD, DQ]	MMX 등록기로부터 높은 자리바이트, 단어 혹은 배단어를 병렬로 풀기
	PUNPCKL [BW, WD, DQ]	MMX 등록기로부터 낮은 자리바이트, 단어 혹은 배단어를 병렬로 풀기
론리연산	PAND	64bit 비트별 론리곱하기
	PNDN	64bit 비트별 론리곱하기부정
	POR	64bit 비트별 론리더하기
	PXOR	64bit 비트별 안맞음론리더하기
자리옮김	PSLL [W, D, Q]	MMX 등록기에서 규정된 값 혹은 직접값으로 조임형단어, 배단어 혹은 4 배단어의 병렬론리왼쪽밀기
	PSKL [W, D, U]	조임형단어, 배단어 혹은 4 배단어의 병렬론리오른쪽밀기
	PSRA [W, D]	조임형단어, 배단어 혹은 4 배단어의 병렬산수오른쪽밀기
자료전송	MOV [D, Q]	MMX 등록기에로 또는 MMX 등록기로부터 배단어 혹은 4 배단어를 이동
상태 Mgt	EMMS	빈 MMX 상태

※명령이 다중자료형[바이트(B), 단어(W), 배단어(D), 4 배단어(Q)]을 지원하는 경우 자료형을 괄호( )안에 표현하였다.

새로운 명령모임의 한가지 특수한 특징은 포화연산의 모임이다. 보통 부호 없는 연산에서 자리넘침이 일어 나면 맨 웃자리비트밖으로 나가는 초파비트들은 잘라 버리게 된다. 이것을 포장둥그리기(wraparound)라고 한다. 이와 같이 하면 자르기영향으로 두 입력연산수보다 더 작은 더하기결과를 낳을수도 있다. 이제 16진수 F000h, 3000h로 표현된 두 단어의 더하기를 고찰해 보자.

F000h = 1111 0000 0000 0000
\+ 3000h = 1111 0000 0000 0000
10010 0000 0000 0000 = 2000h

만일 이 두수가 영상의 세기를 표현한다면 그때 더하기결과는 두 어두운 그림자의 결합이 개개의 어두운 그림자보다 더 밝아 지게 한다. 이것은 전형적인 생각밖의 일이다. 포화연산을 하면 더하기결과에 자리넘침이 생기거나 덜기결과로 내리자리넘침이 발생하는 경우 그 결과가 표현할수 있는 값들가운데서 가장 큰 수 혹은 가장 작은 수로 설정된다. 앞의 실례에 포화연산을 적용하면 다음과 같이 된다.

F000h = 1111 0000 0000 0000
\+ 3000h = 0011 0000 0000 0000
10010 0000 0000 0000
1111 1111 1111 1111 = FFFFH

공통적으로 화상응용은 색이 점차 희미해 지거나 색이 점차 선명해 지는 영향을 받는다. 이것은 한 장면이 점차 다른것으로 되여 버리는 결과를 초래한다. 그리하여 두개의 화상을 무게평균으로 합성한다.

$$\text{Result_Pixel} = \text{A_pixel} \times \text{fade} \times \text{B_pixel} \times (1-\text{fade})$$

이 계산은 A 화상과 B 화상에 있는 매 화소자리에 대하여 수행된다. 영상틀의 계렬이 1로부터 0까지의 색세기값(8bit의 옹근수로써 알맞게 척도화되였다.)을 점차적으로 변화시키면서 얻어 진다면 그 결과는 화상 A로부터 화상 B로 색이 점차 변하게 한다.

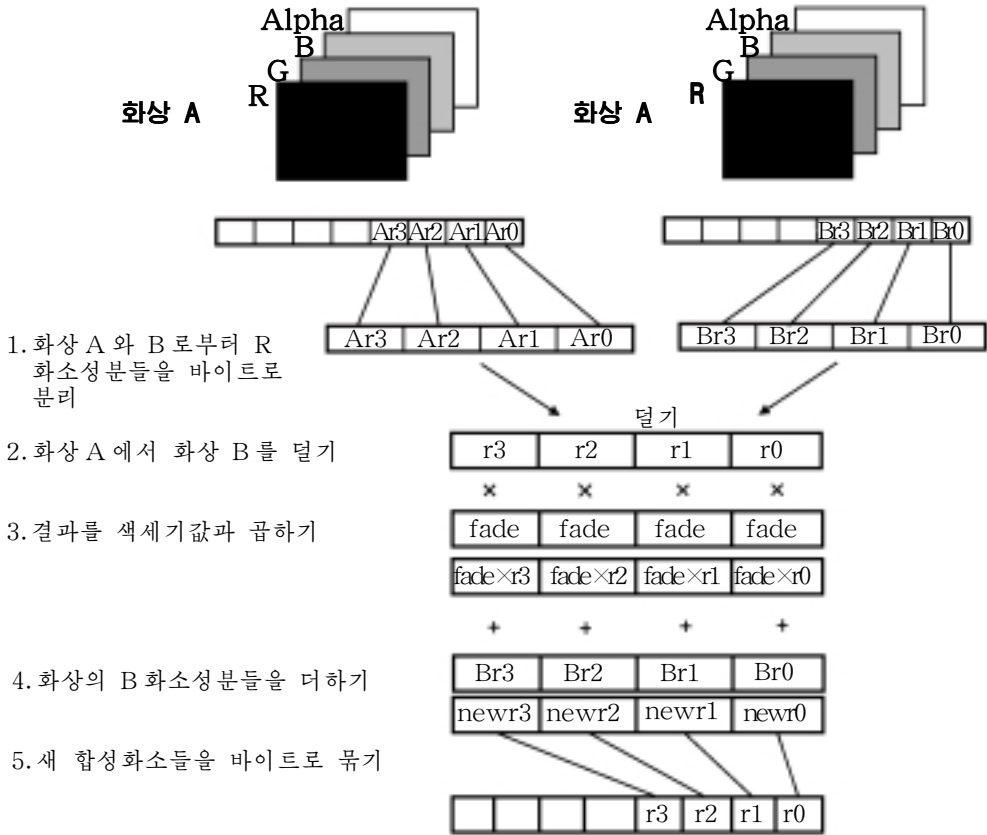
그림 9-10은 하나의 화소모임을 만드는 단계를 순차적으로 보여 주고 있다. 먼저 8bit의 화소성분들이 MMX의 16bit다중능력을 리용할수 있도록 16bit의 요소들로 변환된다. 640×480 분해능을 리용하며 255개의 가능한 색세기값들을 모두 리용한다면 그때 MMX를 리용하여 실행되는 명령의 총수는 535×10^{7}로 된다. MMX명령을 리용하지 않고 이와 같은 계산을 하는 경우에는 1.4×10^{10}개의 명령이 요구된다.

2. PowerPC의 연산종류

PowerPC에도 많은 연산형들이 있다. 표 9-11에서는 이 연산종류를 목록화하여 보여 주었다.

분기지향명령

PowerPC는 보통 무조건적이며 조건적인 분기능력을 가지고 있다. 조건분기명령들은 참인가 거짓인가를 판정하기 위하여 조건등록기의 한비트를 검사하며 연산결과가 0인가 0이 아닌가를 판정하기 위하여 계수등록기의 내용을 검사한다. 조건분기명령에서는 9가지 조건들을 리용한다. 계수등록기의 내용이 검사되면 그 등록기의 내용이 검사하기 전보다 1만큼 감소한다. 이것은 반복순환을 형성하는데서 매우 편리하게 리용된다.

분기명령은 또한 분기명령다음의 위치주소가 련결등록기에 있다는것을 지적할수 있다. 련결등록기에 대하여서는 제 13 장에서 고찰한다. 이 등록기는 호출 및 복귀처리를 쉽게 해준다.

이 연산을 수행하는 MMX 코드렬:

```
pror        mm7, mm7      ;mm7 을 령으로
movq        mm3, fad_val  ;4 배로 개선된 색세기값을 넣기
movd        mm0, imageA   ;화상 A 에서 4 개의 붉은색화소넣기
movd        mm1, imageB   ;화상 B 에서 4 개의 붉은색화소넣기
punpckblw   mm0, mm7      ;4 의 화소를 16bit 로 분리
punpckblw   mm1, mm7      ;4 의 화소를 16bit 로 분리
psubw       mm0, mm1      ;화상 A 에서 화상 B 를 덜기
pmulhw      mm0, mm3      ;덜기결과를 색세기값과 곱하기
padddw      mm0, mm1      ;결과를 화상 B 와 더하기
packuswb    mm0, mm7      ;16bit 결과를 거꾸로 바이트단위로 묶기
```

그림 9-10. 색평면우에서 화상들의 혼합[PELE97]

넣기 및 기억명령

PowerPC 기본방식에서는 넣기 및 기억명령들만이 기억기위치를 호출하며 산수론리연산명령들은 등록기에 의해서만 수행된다. 이것은 축소명령모임콤퓨터(RISC)의 특성이며 이에 대하여서는 제 2 장에서 구체적으로 고찰한다.

표 9-11. PowerPC 의 연산종류(대표적인 연산실례를 포함)

명 령	설 명
	분 기 지 향
b	무조건분기
b1	목적주소로 이행하며 련결등록기에 분기명령다음의 명령의 유효주소를 배치한다.
bc	계수등록기나 조건등록기의 비트에 따르는 조건분기
sc	조작체계봉사를 바라는 체계호출
trap	두 연산수를 비교하여 조건이 만족되면 체계내부새치기조종기를 요구한다.
	넣 기 및 기 억
lwau	단어넣기와 왼쪽에 령을 확장. 원천등록기를 갱신한다.
ld	배단어넣기
lmw	여러단어 넣기.목표등록기로부터 일반목적등록기 31 까지의 등록기들에 련속적으로 단어를 넣기
lswx	목적등록기로부터 시작하는 등록기들에 바이트렬을 넣기. 등록기당 4byte, 등록기 31 로부터 등록기 0 까지 포장둥그리기
	옹 근 수 연 산
add	두 등록기내용을 더하여 세번째 등록기에 넣기
subf	두 등록기내용을 덜어서 세번째 등록기에 넣기
muliw	두 등록기의 낮은 자리 32bit 내용을 곱하여 세번째 등록기에 64bit 의 적으로 넣기
divd	두 등록기의 64bit 내용을 나누어 세번째 등록기에 상을 넣기
	론 리 및 밀 기
cmp	두 연산수를 비교하여 규정된 조건등록기마당에 4 개의 조건비트를 설정
crand	조건등록기 AND. 조건등록기의 두 비트가 론리곱하기하여 결과를 두 비트자리중 하나에 배치
and	두 등록기의 내용을 론리곱하기하고 제 3 의 등록기에 보관
cntlzd	원천등록기의 비트 0 에서부터 시작하여 련속되는 0 의 수를 계수하여 목적등록기에 계수값을 배치
rldic	배단어 등록기의 왼쪽 회전.마스크를 가진 론리곱하기를 취하여 목적등록기에 넣기
sld	원천등록기의 비트들을 왼쪽 밀기하여 목적등록기에 기억
	류 동 소 수 점 수
lfs	기억기로부터 32bit 류동소수점수를 넣어 64bit 형식으로 바꾸고 류동소수점수등록기에 기억
fadd	두 등록기의 내용을 더하여 제 3 의 등록기에 넣기
fmadd	두 등록기의 내용을 곱하고 제 3 의 등록기내용을 더하여 제 4 등록기에 결과를 보관
fcmpu	두 류동소수점연산수를 비교하여 조건비트들을 설정
	캐 쉬 관 리
dcbf	자료캐쉬블로크지우기. 특별한 목표주소에서 캐쉬를 검사하고 지우기 연산을 수행
icbi	캐쉬블로크무효명령

넣기 및 기억명령을 규정하는 두가지 특징들은 다음과 같다.

- **자료크기:** 자료는 바이트, 반단어, 단어 혹은 배단어의 단위로 전송될수 있다.명령들은 다중등록기에로 혹은 다중등록기로부터 바이트렬을 넣거나 기억하는데 리용할수 있다.
- **부호확장:** 반단어 혹은 단어넣기에서는 64bit 의 목적등록기의 왼쪽에 있는 리용하지 않는 비트들을 령 혹은 넣은 량의 부호비트로 채운다.

제 6 절. 아쎔블리언어

CPU 는 기계명령을 리해하고 실행할수 있다. 이러한 명령들은 단순히 콤퓨터에 기억된 2 진수들이다. 프로그람작성자가 직접 기계어로 프로그람을 작성하려고 하는 경우에는 2 진자료를 가지고 프로그람을 작성해야 한다.

이제 단순한 BASIC 명령문을 고찰해 보자.

N=I+J+K

이 명령문을 기계어로 작성하고 I, J 및 K 에 값 2, 3, 4 를 할당한다고 하자. 결과를 그림 9-11 ㄱ에 보여 주었다. 프로그람은 16 진수로 표현된 위치 201 에서 시작한다. 이 프로그람은 다음과 같은 4 개의 명령으로 구성된다.

1. 위치 201 의 내용을 AC 에 넣기
2. 위치 202 의 내용을 AC 에 더하기
3. 위치 203 의 내용을 AC 에 더하기
4. AC 의 내용을 위치 204 에 기억

우의 과정은 오랜 시간이 걸리며 오유를 자주 발생하게 되는 과정으로 된다.

이것을 약간 개선한것이 2 진표기법대신에 16 진표기법으로 프로그람을 작성하는것이다(그림 9-11 ㄴ). 여기서는 행의 렬로써 프로그람을 쓰며 매행에는 기억기위치주소와 그 위치에 기억되는 2 진값의 16 진부호가 들어 간다. 이 16 진프로그람을 콤퓨터에 입력시켜 매행을 2 진수로 변환하고 지정된 위치에 그것을 기억한다.

이 16 진프로그람보다 더 개선된것은 매 명령에 기호이름을 붙여 리용하는것이다. 이것을 리용하면 그림 9-11 ㄷ에서 볼수 있는것과 같은 **기호프로그람**을 작성할수 있다. 프로그람에서 입력의 매행은 역시 기억기위치를 표현한다. 매행은 공간적으로 서로 분리되여 있는 세개의 마당으로 구성된다. 제일 첫 마당은 기억기위치주소를 표현한다. 두번째 마당은 조작코드 즉 연산종류를 식별해 주는 세개의 글자기호로 되여 있다. 기억기참조명령인 경우는 세번째 마당에 주소를 쓴다. 어떤 위치에 임의의 자료를 축적하기 위하여 기호 DAT 로 표현되는 **의사명령**을 리용한다. 이 명령은 행에서 세번째 마당이 첫 마당에서 규정한 위치에 기억되는 16 진수를 포함한다는것을 지시한다.

이와 같은 형태의 입력을 위하여서는 약간 더 복잡한 프로그람이 있어야 한다. 프로그람은 입력의 매행을 접수하여 두번째 및 세번째마당을 기초로 2 진수를 발생시키며 그것을 첫 마당에서 규정한 위치에 기억해야 한다.

기호프로그람을 리용하면 프로그람을 보다 쉽게 작성하지만 다루기 불편한 점이 아직도 적지 않다. 특히 매 단어에 대한 절대주소가 주어 져야 하는데 이것은 프로그람과 자료를 기억기의 한 장소에만 넣을수 있다는것을 의미하며 프로그람을 작성하기전에 그

장소를 알아야 한다는것을 의미한다. 더우기 한 행을 프로그람에 추가하거나 또 한 행을 삭제하려고 하는 경우에는 반드시 그에 뒤따르는 모든 단어들의 주소를 변화시켜야 한다.

주소	내용			
101	0010	0010	0000	0001
102	0001	0010	00000	0010
103	0001	0010	0000	0011
104	0011	0010	0000	0100
201	0000	0000	0000	0010
202	0000	0000	0000	0011
203	0000	0000	0000	0111
204	0000	0000	0000	0000

ㄱ)

101	LDA	201
102	ADD	202
103	ADD	203
104	STA	204
201	DAT	2
202	DAT	3
203	DAT	4
204	DAT	0

ㄷ)

주소	내용
101	2201
102	1202
103	1203
104	3204
201	0002
202	0003
203	0004
204	0000

ㄴ)

표식	연산	연산수
FORMUL	LDA	I
	ADD	J
	ADD	K
	STA	N
I	DATA	2
J	DATA	3
K	DATA	4
N	DATA	0

ㄹ)

그림 9-11. 공식 N=I+J+K의 계산; ㄱ-2진프로그람, ㄴ-16진프로그람, ㄷ-기호프로그람, ㄹ-아쎔블리언어

이로부터 보다 더 좋은 체계 즉 우리가 일반적으로 많이 리용하는 체계들에서는 기호주소를 리용한다. 이와 관련한것을 그림 9-11 ㄹ 에 설명하였다. 매행은 역시 3개의 마당으로 이루어 진다. 첫 마당은 주소마당이지만 여기서는 기호주소를 절대값주소대신 리용한다. 일부 행들에는 주소가 없는데 그것은 그 행들의 주소가 앞행의 주소보다 하나 더 크다는것이 명백히 암시되여 있기때문이다.

이와 같이 구성되는 언어가 바로 **아쎔블리언어**이다. 아쎔블리언어로 작성된 프로그람(아쎔블리프로그람)은 **아쎔블리**에 의하여 기계어로 번역된다. 이 프로그람은 앞에서 이미 론의한 기호변환뿐아니라 기호주소에 일정한 형의 기억기주소를 할당하기도 한다.

아쎔블리언어의 개발은 콤퓨터기술의 진화발전에 큰 영향을 주었다. 이것은 현재 널리 리용되고 있는 고급언어의 첫 단계였다. 일부 프로그람작성자들만이 아쎔블리언어를 리용하고 있지만 모든 콤퓨터들은 아쎔블리언어로 프로그람을 작성할수 있게 되여 있다. 아쎔블리언어는 번역프로그람 혹은 콤파일러나 I/O 루틴과 같은 체계프로그람개발에 많이 리용된다.

참고문헌

기계어 및 명령모임설계와 관련하여 좋은 책들로는 [HENN96], [TANE90], [HAYE88]을 비롯하여 여러개가 있다. Pentium 명령모임은 [BREY97]과 [DEWA90]을, PowerPC 명령모임은 [IBM94]와 [WEIS94]를 참고하면 잘 알수 있다.

BREY97 Brey, B. *The Intel Microprocessors: 8086/8066, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Processor.* Upper Saddler River, NJ: Prentice Hall, 1997.

DEWA90 Dewar, R., and Smosna, M. *Microprocessors: A Programmer's View*. New York: McGraw-Hill, 1990.

HAYE88 Hayes, J. *Computer Architecture and Organization, 2nd edition.* New York: McGraw-Hill, 1988.

HENN96 Hennessy, J., and Patterson, D. *Computer Architecture: A Quantitative Approach.* San Mateo, CA: Morgan Kaufmann, 1996.

IBM94 International Business Machines, Inc. *The PowerPC Architecture : A Specification for a New Family of RISC Processors.* San Francisco, CA: Morgan Kaufmann, 1994.

TANE90 Tanenbaum, A. *Structured Computer Organization.* Englewood Cliffs, NJ: Prentice Hall, 1990.

WEIS94 Weiss, S., and Smith, J. *POWER and PowerPC.* San Francisco: Morgan Kaufmann, 1994.

련습문제

1. 많은 CPU 들은 조임형 10 진수로 산수연산을 진행한다. 10 진연산규칙들은 2 진연산규칙들과 류사하지만 2 진론리가 리용되는 경우에는 10 진결과의 개별적인 수자들을 일부 수정해야 한다.

   두 부호 없는 수들의 10 진더하기를 고찰해 보자. 매수가 N개의 수자로 구성된다면 매수에 $4N$개의 비트들이 있게 된다. 이 두수가 2 진가산기에서 더해 진다. 이와 같은 경우에 결과를 수정하는 단순한 규칙을 제기하시오. 두수 1698 과 1786 에 대하여 우의 형식대로 더하기를 진행하시오.

2. 10 진수 X의 10 의 보수는 $10N-X$로 정의된다.

   여기서 N은 10 진수에서 수자들의 개수이다. 10 진덜기를 수행하는 경우 10 의 보수표현법의 리용에 대하여 설명하시오. $(0736)_{10}$에서 $(0326)_{10}$을 더는 틀을 설명하시오.

3. 다음의 계산을 진행하는 프로그람을 작성하고 그것을 0, 1, 2 및 3 주소를 리용하는 콤퓨터들과 결부시켜 비교하시오.

$$X = (A + B \times C)/(D - E \times F)$$

   프로그람작성에서 리용할수 있는 명령들은 아래와 같다.

0 주소	1 주소	2 주소	3 주소
PUSH M	LOAD M	MOVE (X ← Y)	MOVE(X ← Y)
POP M	STORE M	ADD(X ← X × Y)	ADD(X ← X + Y)
ADD	ADD M		
SUB	SUB M	SUB(X ← X − Y)	SUB(X ← X − Z)
MUL	MUL M	MUL(X ← X × Y)	MUL(X ← X × Z)
DIV	DIV M		

4. 2개의 n비트명령으로 된 명령모임을 가진 가상적인 콤퓨터를 생각하자. 첫 비트는 조작코드를 규정하며 나머지비트들은 주기억기의 2^n-1 개의 n 비트단어들중 하나를 규정한다. 두개의 명령은 다음과 같다.

   SUBS X : 축적기로부터 위치 X 의 내용을 덜어서 위치 X 와 축적기에 결과를 기억한다.

   JUMP X : 프로그람계수기에 주소 X를 넣는다.

   주기억기의 단어는 명령이나 2 의 보수표기법으로 표현된 2 진수이다. 다음의 연산을 어떻게 프로그람화하는가를 규정하면 이 명령저장고가 합리적으로 완성된다는 것을 론증하시오.

   ㄱ. 자료전송: 위치 X를 축적기에, 축적기를 위치 X로 이동
   ㄴ. 더하기: 위치 X의 내용을 축적기에 더하기
   ㄷ. 조건분기
   ㄹ. 론리더하기
   ㅁ. I/O 연산

5. 대다수 명령모임들은 아무러한 연산도 수행하지 않는 명령 NOOP를 가지고 있다. 이 명령은 프로그람계수기를 증가시킬뿐 CPU 상태에는 영향을 주지 않는다. 이 명령의 리용실례를 만들어 보시오.

6. CPU 가 틀의 호출 및 복귀를 관리하는데 탄창을 리용한다고 하자. 이때 프로그람계수기처럼 탄창의 꼭대기를 리용하면 프로그람계수기가 없어도 되겠는가?

7. 부록 9-1 에서 탄창이 틀의 조종과 같은 목적을 위하여 CPU 에 의해서만 리용되는 경우에는 명령모임에 탄창지향명령이 없다는것을 보여 주고 있다. CPU 가 탄창지향명령이 없이 임의의 목적에 탄창을 어떻게 리용할수 있는가?

8. 다음식들을 뒤배치표기법으로부터 사이배치표기법으로 바꾸시오.

   ㄱ. AB + C + D
   ㄴ. AB/CD/+
   ㄷ. ABCDE +××/
   ㄹ. ABCDE + F/+ G − H/×+

9. 다음식들을 사이배치표기법으로부터 뒤배치표기법으로 바꾸시오.

   ㄱ. A + B + C + D + E
   ㄴ. (A + B)×(C + D)+ E
   ㄷ. (A × B)+(C × D)+ E
   ㄹ. (A − B)×(((C − D × E)/F)/G)× H

10. 식 A + B + C 를 디직스트러(Dijkstra)의 알고리듬을 리용하는 뒤배치표기법으로 바꾸시오. 그리고 이에 동반되는 계단들을 보여 주시오. 결과가 (A+B)−C 혹은 A+(B−C)와 같은가?

11. Pentium Ⅱ기본방식은 더하기연산후에 10 진보정을 하는 명령 DAA 를 가지고 있다. DAA 는 다음과 같은 순서로 명령들을 수행한다.

```
If (( AL AND 0FH) > 9 ) OR (AF = 1) then
    AL ← AL + 6;
    AF ←1;
else
    AF ← 0;
```

```
endif
If (AL > 9FH) OR (CF = 1) then
      AL ← AL + 60H;
      CF ← 1;
                  else
      AF ← 0;
endif
```

《H》는 16 진수를 의미한다. AL 은 두개의 부호 없는 8bit 등록기들의 더하기결과를 가지는 8bit 등록기이다. AF 는 더하기결과의 비트 3 으로부터 비트 4 에 자리올림이 있는 경우에 설정되는 기발이다. CF 는 비트 7 에서 비트 8 로 자리올림이 있는 경우에 설정되는 기발이다. DAA 명령에 의해 수행되는 기능을 설명하시오.

12. Pentium Ⅱ의 비교명령(CMP)은 목적연산수에서 원천연산수를 던다. 결과로써 상태기발 (C, P, A, Z, S, O)을 변화시키지만 연산수의 내용은 변화시키지 않는다. 이 CMP 명령에는 조건이행(Jcc) 혹은 모임조건 (SETcc)명령이 뒤따르게 된다. 여기서 cc 는 표 9-9 에서 목록화한 16 개 조건들중의 하나이다. 부호 없는 수들의 비교에서 론의한 조건들이 정확하다는것을 론증하시오.

13. 대다수 극소형처리장치명령모임들은 조건을 검사하여 그 조건이 맞는 경우에는 목적연산수를 설정하는 명령을 가지고 있다. 그 실례가 바로 Pentium Ⅱ의 SETcc, Motorola MC68000 의 Scc, National NS32000 의 Scond 이다.

ㄱ. 이 명령들사이에는 약간의 차이가 있다.

- SETcc 와 Scc 는 바이트에 대한 연산만 하며 Scond 는 바이트, 단어 ,배단어에 대한 연산을 한다.
- SETcc 와 Scond 는 참인 경우 옹근수1로 연산수를 설정하고 거짓인 경우에는 령으로 설정한다. Scc 는 참이면 모든 비트가 1 인 바이트를 설정하며 거짓이면 모든 비트가 0 인 바이트를 설정한다. 이러한 서로 다른 기능의 우점과 결함은 무엇인가?

ㄴ. 이 명령들에서는 임의의 조건코드기발을 설정하지 않으므로 명령결과의 명시적인 시험은 그의 값을 결정할것을 요구하게 된다. 조건코드들이 이 명령결과로 설정되여도 되는가를 론의하시오.

ㄷ. IF a > b THEN 과 같은 단순 IF 명령문은 불값을 만드는 수표현법을 리용하여 실현할수 있다. 이 표현법은 프로그람의 도달점에서의 불식의 값을 표현하는 **흐름조종**방법과는 상반된다. 콤파일러는 다음과 같은 80x86 코드로 된 IF a>b THEN 을 번역한다.

```
              SUB      CX,CX       ; 등록기 CX 를 0 으로 설정
              MOV      AX,B        ; 위치 B 의 내용을 등록기 AX 로 이동
              CMP      AX,A        ; 등록기 AX 의 내용과 위치 A 를 비교
              JLE      TEST        ; A ≦ B 이면 건너뛰기
              INC      CX          ; 등록기 CX 의 내용에 1 을 더하기
TEST          JCXZ     OUT         ; CX 의 내용이 령이면 이행
THEN

OUT
```

(A>B)의 결과는 등록기에 보존된 불값이며 이 결과는 이 문제에서 보여 준 코드

흐름만이 아니라 다른 프로그람에서도 리용할수 있다. 이를 위하여서는 등록기 CX를 리용하는것이 편리한데 그것은 많은 분기 및 순환고리조작코드들이 고리내에서의 시험에 CX 를 리용하기때문이다. 기억기와 실행시간을 단축하는 SETcc 명령을 리용하여 이 프로그람을 다시 작성하시오(*암시:* SETcc 명령외에 다른 새로운 80x86 명령들은 필요 없다.).

ㄹ. 다음과 같은 고급언어명령문이 있다고 하자.

A: = (B > C) OR (D = F)

콤파일러는 이 명령문을 다음과 같은 코드로 만든다.

```
              MOV     EAX, B      ; 위치 B의 내용을 등록기 EMX에로 이동
              CMP     EAX, C      ; 등록기 EAX의 내용과 위치 C를 비교
              MOV     BL, O       ; 0이 거짓을 표현한다.
              JLE     N1          ; B≦C이면 이행
              MOV     BL, 1       ; 1이 거짓을 표현
N1            MOV     EAX, D
              MOV     EAX, F
              MOV     BH, O
              JNE     N2
              MOV     BH, 1
N2            OR      BL, BH
```

기억기와 실행시간을 절약하는 SETcc 명령을 리용하여 이 프로그람을 다시 작성하시오.

14. 부록 9-1 에서 고찰한 사이배치표기를 뒤배치표기로 바꾸는 알고리듬을 리용하여 그림 9-15 의 식을 뒤배치표기법으로 바꾸는 단계들을 보여 주시오. 그림 9-17 과 류사한 표현을 리용하시오.

15. 그림 9-16 과 같은 표현법을 리용하여 그림 9-17 에 있는 식을 계산하시오.

16. 그림 9-18 에서 작은끝지면배치도를 큰끝배치도와 같이 번호가 붙도록 다시 그리시오. 64bit 행들로 된 기억기를 왼쪽에서 오른쪽으로, 우에서 아래로 바이트들을 목록화하여 보여 주시오.

17. 같은 자료구조에서 큰끝와 작은끝지면배치도를 그리시오. 이때 그림 9-18 의 형식과 그 결과에 대한 설명을 리용하시오.

ㄱ.
```
strcut {
        double  i ;   //0x1112131415161718
} s1;
```

ㄴ.
```
struct {
        int  i;       //0x11121314
        int  j;       //0x15161718
} s2;
```

ㄷ.
```
struct {
        short  i ;    //0x1112
        short  j ;    //0x1314
        short  k;     //0x1516
        short  l ;    //0x1718
  } s3;
```

18. PowerPC 구성방식명세서는 처리장치가 작은끝방식을 어떻게 실현해야 하는가를 보여 주지 않는다. 구성방식명세서는 작은끝방식에서 동작할 때 처리장치가 가져야 할 기억기에 대한것만을 규정하고 있다. 큰끝으로부터 작은끝으로 자료구조를 변화시키는 경우 처리장치는 바이트교체기구를 자유롭게 실현하거나 일부 주소변경기구를 자유롭게 리용한다. 현재 PowerPC 처리장치들은 모두 큰끝방식을 암시적으로 리용하며 작은끝방식으로 자료를 취급하는 경우에는 주소변경기구를 리용한다.
    이제 그림 9-18 에서 정의한 구조 s를 고찰해 보자. 이 그림의 오른쪽에 있는 지면배치도는 처리장치에서 참고하는 구조 s를 보여 주고 있다. 구조 s가 작은끝방식으로 번역되는 경우 사실상 기억기배치도는 그림 9-12 와 같다. 그림 9-12 를 설명하고 이것이 쉬운 기억기배치방법이라는것과 이 방법의 효과성을 론증하시오.

작은 마무리 주소배치

바이트 주소								
00	00	01	02	03	11 04	12 05	13 06	14 07
08	21 08	22 09	23 0A	24 0B	25 0C	26 0D	27 0E	28 0F
10	'D' 10	'C' 11	'B' 12	'A' 13	31 14	32 15	33 16	34 17
18	18	19	51 1A	52 1B	1C	'G' 1D	'F' 1E	'E' 1F
20	20	21	22	23	61 24	62 25	63 26	64 27

그림 9-12. 기억기에서 Power PC 의 작은 마무리구조

19. 콤퓨터의 끝배치성을 결정하는 하나의 작은 프로그람을 작성하고 그 결과를 출력하시오.

부록 9-1. 탄창

1. 탄창

탄창은 한번에 하나의 요소에만 호출할수 있는 요소들의 질서정연한 모임이다. 여기서 탄창의 호출점을 탄창의 꼭대기라고 한다. 탄창의 요소수 혹은 탄창의 **길이**는 변한다. 항목들은 탄창의 꼭대기를 통해서만 탄창에 추가되거나 탄창으로부터 제거된다. 이런 리유로부터 탄창을 **아래로 밀어넣기목록** 혹은 **후입선출** (LIFO: Last IN-First Out)목록이라고 할수 있다.

그림 9-13 은 기본탄창연산을 보여 주고 있다. 탄창이 일정한 수의 요소를 포함할 때에는 일정한 수의 점에서 시작한다. PUSH 연산은 탄창의 꼭대기에 하나의 새로운 항목을 덧붙이며 POP연산은 탄창으로부터 꼭대기항목을 제거한다. 두 경우에 탄창의 꼭대기가 뒤따라 움직인다. 2개의 연산수를 리용하는 2진연산(곱하기, 나누기, 더하기, 덜기)은 연산수로서 꼭대기 2 개의 탄창항목을 리용하며 이 두 항목을 꺼내고 결과를 탄창에 밀어 넣는다. 오직 하나의 연산수만을 가지는 연산(론리부정)은 탄창의 꼭대기에 있는 항목을 리용한다. 이러한 모든 연산들은 표 9-12 에 개괄하여 보여 주었다.

2. 탄창실현

탄창은 CPU 실현의 한 구성부분으로 되는 쓸모 있는 기억기구조이다. 이미 제 4 절에서 론의한 탄창의 한가지 리용은 틀의 호출 및 복귀를 관리하는것이였다. 탄창은 프로그람작성자에게도 효과적으로 리용된다. 이에 대한 실례는 이 절에서 후에 론의하는 식평가에서 보기로 하자.

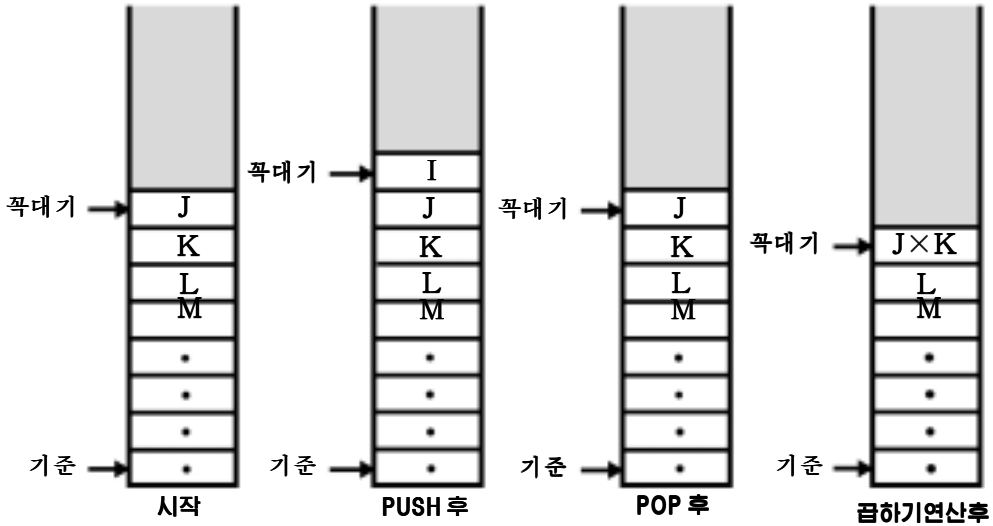



그림 9-13. 기본탄창동작

표 9-12. 탄창지향연산

연산종류	설명
PUSH	탄창꼭대기에 새로운 요소를 덧붙인다.
POP	탄창의 꼭대기요소를 제거한다.
일원연산	탄창의 꼭대기요소에 대한 연산을 수행한다. 꼭대기요소를 결과로 재배치한다.
2 진연산	탄창의 꼭대기 두 요소에 대한 연산을 수행한다. 탄창의 꼭대기 두 요소는 제거된다. 연산결과를 탄창의 꼭대기에 배치한다.

탄창의 실현은 그의 잠재적인 능력을 리용하는데서 한 부분에 지나지 않는다. 프로그람작성자가 리용할수 있는 탄창연산을 만들자면 PUSH, POP 와 같은 탄창지향명령들과 연산수로써 꼭대기 하나 혹은 두개의 탄창요소를 리용하는 연산명령들이 있어야 한다. 이 모든 연산들은 탄창의 꼭대기라고 하는 동일한 위치를 리용하므로 연산수 혹은 연산수들의 주소가 암시적이며 따라서 이것은 명령에 포함시킬 필요가 없다. 이 명령들은 제 9 장 제 1 절에서 론의한 0 주소명령들이다.

틀의 조종과 같은 목적으로 탄창을 오직 CPU 에 의해서만 리용한다면 명령모임에는 절대적인 탄창지향명령이 없을것이다. 이 경우에도 탄창실현은 탄창요소들을 기억하는데 리용되는 기억기위치들의 모임이 있을것을 요구한다. 이와 같은 방법을 그림 9-14 ㄱ에서 보여 주고 있다. 그림에서 보는바와 같이 위치들의 블로크가 탄창용으로 주기억기(혹은

가상기억기)에 예약된다. 이 블로크에는 보통 부분적으로 탄창요소들이 차 있게 되며 나머지는 탄창성장에 리용할수 있다.

세개의 주소가 알맞는 연산을 위하여 필요하게 되며 이 주소들은 보통 CPU 등록기들에 기억된다.

- **탄창지시기:** 탄창의 꼭대기주소를 가진다. 항목이 탄창에 추가되거나 탄창으로부터 제거되면 지시기는 탄창의 새로운 꼭대기주소를 가지도록 증가되거나 감소된다.
- **탄창기준:** 예약된 블로크에서 밑바닥위치의 주소를 가진다.
- **탄창한계:** 예약된 블로크의 다른끝주소를 가진다.

블로크가 탄창용으로 모두 리용되였다면 이때 시도되는 PUSH 명령은 오유를 발생한다. 일반적으로 오늘날 대부분의 콤퓨터들에서는 탄창기준이 예약된 탄창블로크의 제일 높은 주소로 되며 그 한계는 낮은 주소끝에 있게 된다. 따라서 탄창은 보다 높은 주소로부터 보다 낮은 주소로 성장한다.

탄창연산의 속도를 높이기 위하여 꼭대기의 두 탄창요소들을 그림 9-14 ㄴ에서 보는 바와 같이 등록기들에 기억한다. 이 경우 탄창지시기는 탄창의 세번째 요소의 주소를 가진다.

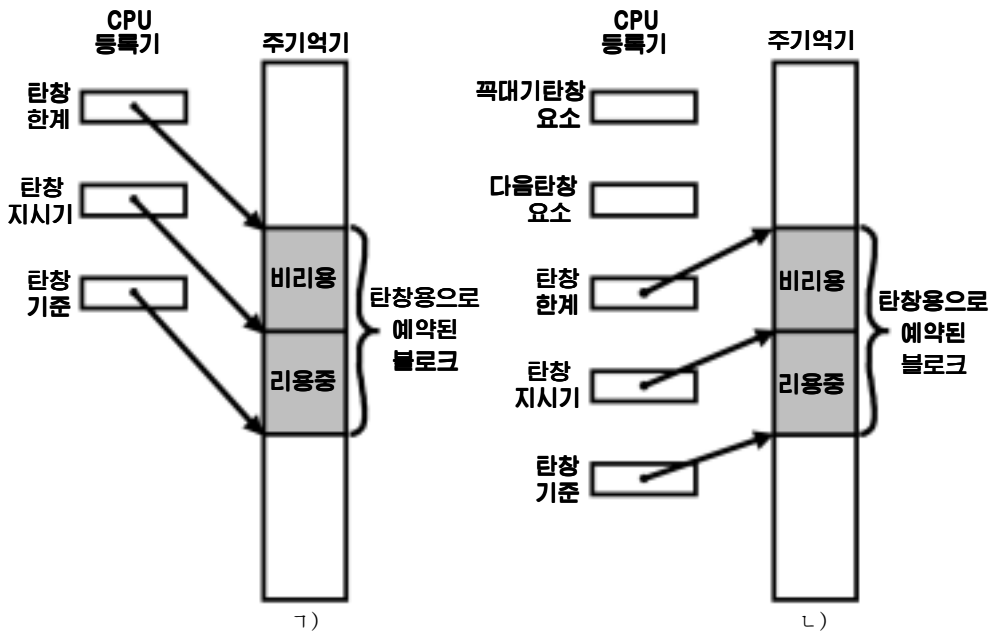



그림 9-14. 일반적인 탄창구성
ㄱ-기억기내의 전체 탄창, ㄴ-등록기로 된 2개의 꼭대기요소

3. 식평가

수학식들은 보통 **사이배치**표기법으로 표현된다. 이 양식에서는 연산자가 연산수들사이에 끼우게 된다. 복잡한 식들의 표현에서는 식의 평가순위를 결정하기 위하여 괄호를 리용한다. 실례로 $a+(b \times c)$는 $(a+b) \times c$와 다른 결과를 준다. 괄호를 매우 적게 리용

하기 위하여 연산은 보통 암시적인 우선권을 가진다. 일반적으로 곱하기는 더하기보다 높은 우선권을 가지며 따라서 a+b × c 는 a+(b × c)와 같은 결과를 낳는다.

사이배치표기법과 대치되는 기술은 **역방향뽈스까**(reverse Polish) 혹은 **뒤배치**(postfix)표기법이다. 이 표기법에서는 연산자가 두 연산수의 뒤에 놓인다. 실례를 들면 다음과 같다.

ab+ ← a+b
abc×+ ← a+(b×c)
ab+c× ← (a+b)×c

보는바와 같이 식의 표현이 복잡하지만 이 표기법을 리용할 때에는 우선권이 필요 없다. 표기법의 우점은 이 양식으로 표현된 식이 탄창을 리용하여 쉽게 평가된다는것이다. 뒤배치표기법으로 표현된 식은 왼쪽에서 오른쪽으로 주사된다. 식의 매 요소들에는 다음과 같은 규칙이 적용된다.

1. 요소가 변수 혹은 상수인 경우 그것을 탄창에 밀어 넣는다.
2. 요소가 연산자이면 탄창의 꼭대기 두개의 항목을 꺼내여 연산을 하고 그 결과를 밀어 넣는다.

전체 식이 주사되면 결과는 탄창의 꼭대기에 있게 된다.

	탄창	일반등록기	단일등록기
	Push a	Load G[1], a	Load d
	Push b	Subtract G[1], b	Multiply e
	Subtract	Load G[2], d	Add c
	Push c	Multiply G[2], e	Store f
	Push d	Add G[2], c	Load a
	Push e	Divide G[1], G[2]	Subtract b
	Multiply	Store G[1], f	Divide f
	Add		Store f
	Divide		
	Pop f		
명령의 개수	10	7	8
기억기호출	10 op + 6 d	7 op + 6 d	8 op + 8 d

그림 9-15. f=(a−b)/(c+d × e)의 계산을 위한 세 프로그람의 비교

이 알고리듬은 식의 평가에 매우 편리한 알고리듬으로 된다. 따라서 많은 콤파일러들이 고급언어의 식을 뒤배치표기법으로 바꾼 다음 기계명령을 만들어 낸다. 그림 9-15는 탄창지향명령을 리용하여 식 f=(a−b)/(c+d×e)을 평가하기 위한 기계명령렬을 보여 주고 있다. 이 그림은 또한 1주소명령과 2주소명령의 리용에 대해서도 보여 준다. 탄창지향규칙이 마지막 두 경우에는 리용되지 않았지만 뒤배치표기법은 기계명령을 발생시키기 위한 안내자로서의 역할을 하고 있다. 탄창프로그람에서 사건들의 순서를 그림 9-16에 보여 주었다.

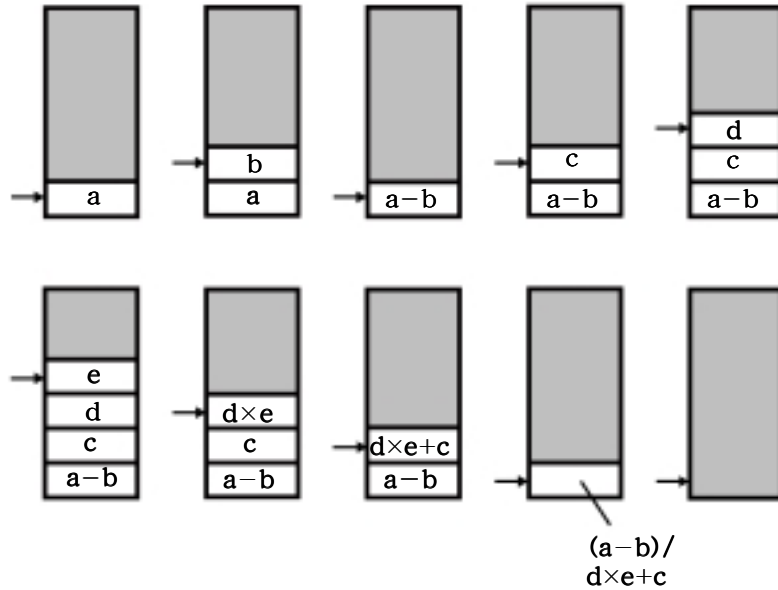



그림 9-16. f = (a−b)/(d×e+c)의 계산에서 탄창의 리용

사이배치식을 뒤배치식으로 바꾸는 처리는 탄창을 리용하면 매우 쉽다. 다음의 알고리듬은 Dijkstra 가 제안한것이다[DIJK63]. 알고리듬은 사이배치식을 왼쪽에서 오른쪽으로 가면서 주사하여 뒤배치식으로 전개하며 주사하면서 출력한다.

입력	출력	탄창
A+B×C+(D+E)×F	empty	empty
+B×C+(D+E)×F	A	empty
B×C+(D+E)×F	A	+
×C+(D+E)×F	AB	+
C+(D+E)×F	AB	+×
+(D+E)×F	ABC	+×
(D+E)×F	ABC×+	+
D+E)×F	ABC×+	+(
+E)×F	ABC×+D	+(
E)×F	ABC×+D	+(+
)×F	ABC×+DE	+(+
×F	ABC×+DE+	+
F	ABC×+DE+	+×
empty	ABC×+DE+F	+×
empty	ABC×+DE+F×+	empty

그림 9-17. 사이배치식을 뒤배치식으로 변환

단계는 다음과 같다.

1. 입력에서 다음요소를 조사한다.
2. 연산수이면 그것을 출력한다.
3. 여는 괄호이면 탄창에 밀어 넣는다.
4. 연산자인 경우에는

- 탄창의 꼭대기가 여는 괄호이면 연산자를 밀어 넣는다.
- 탄창의 꼭대기보다 더 높은 우선권을 가지면(곱하기와 나누기는 더하기와 덜기보다 더 높은 우선권을 가진다.) 연산자를 밀어 넣는다.
- 기타는 탄창에서 연산자를 꺼내여 단계 4를 반복한다.

5. 닫는 괄호이면 여는 괄호를 만나게 될 때까지 출구로 연산자를 꺼낸다. 꺼내고 여는 괄호를 없앤다.
6. 입구가 더 있으면 단계 1로 간다.
7. 더이상 입구가 없으면 남은 연산수들은 탄창에 넣지 않는다.

그림 9-17에서는 이 알고리듬의 리용을 설명하였다. 이 실례를 통하여 탄창에 기초한 알고리듬의 능력을 어느 정도 알수 있다.

부록 9-2. 작은끝, 큰끝, 쌍끝배치

단어안의 바이트와 바이트안의 비트가 어떻게 참조되며 표현되는가를 리해하는것은 중요한 문제의 하나이다. 부록에서는 먼저 바이트순서화를 고찰하고 그다음 비트순서화를 보기로 한다.

1. 바이트순서화

끝배치성의 개념은 Cohen 의 문헌[COHE81]에서 처음으로 론의되였다. 바이트에 대한 끝배치성은 여러바이트스칼라값들의 바이트순서화와 함께 고찰해야 한다. 고찰은 실례를 통하여 하는것이 제일 좋다. 이제 32bit 의 16 진값 12345678 을 바이트단위의 기억기위치(바이트위치)184 에 32bit 의 단어로 기억하는 경우를 보자.

주소	값
184	12
185	34
186	56
187	78

주소	값
184	78
185	56
186	34
187	12

값은 4byte 로 구성되는데 맨 아래자리(제일 무게가 작은)바이트값이 78 이고 맨 웃자리(제일 무게가 큰)바이트값은 12 이다. 이 값을 기억하는데는 두가지 방법이 있다. 왼쪽에 있는 배치도는 제일 무게가 큰 바이트를 가장 낮은 바이트주소에 넣는다. 이것을 큰끝이라고 하며 왼쪽에서 오른쪽으로의 순서 간단히 좌우순서(left-to-right order)라고도 한다. 오른쪽에 있는 배치도는 제일 무게가 작은 바이트를 가장 낮은 바이트주소에 넣는다. 이것을 작은끝이라고 하며 오른쪽에서 왼쪽으로의 순서 간단히 좌우순서(right-to-left order)라고도 한다.[2] 이와 같은 기억기에로의 바이트배치순서는 산수연산장치에서 산수연산을 위한것이다. 주어 진 여러바이트스칼라값에 대한 큰끝과 작은끝은 서로 바이트가 역전된 배치도이다.

---

[2] 큰끝과 작은끝배치라는 말은 Jonathan Swift 가 쓴 "갈리버의 려행기" 제 1 편 제 4 장에서 유래된것이다. 이 말은 큰 쪽에서 수류탄을 던지는 종족과 작은 쪽에서 수류탄을 던지는 종족사이 종교전쟁을 가리키는 말이다.

끌배치성의 개념은 여러바이트실체를 단일한 주소를 가진 단일한 자료항목으로 취급해야 할 경우에 적용되는 개념이다. Intel 80x86, Pentium Ⅱ, VAX 와 Alpha 와 같은 일부 콤퓨터들은 작은끌방식의 기계이며 IBM System 370/390, Motorola 680x0, Sun SPARC 와 대다수 RISC 기계들은 큰끌방식의 기계들이다. 이러한 사정은 한 끌배치형의 기계로부터 다른 끌배치형의 기계로 자료를 전송할 때 그리고 프로그람작성자가 여러 바이트스칼라안의 개별적인 바이트 혹은 비트들을 조작하려고 하는 경우에 문제가 제기된다.

끌배치성의 속성은 개별적인 자료단위를 벗어 나지 않는다. 임의의 콤퓨터에서 파일, 자료구조, 배렬과 같은 집합체들은 매개가 다 끌배치성을 가지고 여러자료단위로 구성된다. 그러므로 한 격식의 끌배치로부터 다른 격식의 끌배치에로의 기억기블로크의 변환은 자료구조에 대한 지식을 요구한다.

```
Struct{
   int        a;      //0x1112_1314                  word
   int        pad;    //
   double     b;      //0x2122_2324_2526_2728        doubleword
   char*      c;      //0x3132_3334                  word
   char       d[7];   //'A','B','C','D','E','F','G'  byte array
   short      e;      //0x5152                       halfword
   int        f;      //0x6161_6364                  word
 } s;
```

큰끝주소배치

바이트 주소								
00	11 00	12 01	13 02	14 03	04	05	06	07
08	21 08	22 09	23 0A	24 0B	25 0C	26 0D	27 0E	28 0F
10	31 10	32 11	33 12	34 13	'A' 14	'B' 15	'C' 16	'D' 17
18	'E' 18	'F' 19	'G' 1A	1B	51 1C	52 1D	1E	1F
20	61 20	62 21	63 22	64 23				

큰끝주소배치

								바이트 주소
07	06	05	04	11 03	12 02	13 01	14 00	00
21 0F	22 0E	23 0D	24 0C	25 0B	26 0A	27 09	28 08	08
'D' 17	'C' 16	'B' 15	'A' 14	31 13	32 12	33 11	34 10	10
1F	1E	51 1D	52 1C	'A' 1B	'G' 1A	'F' 19	'E' 18	18
				61 23	62 22	63 21	64 20	20

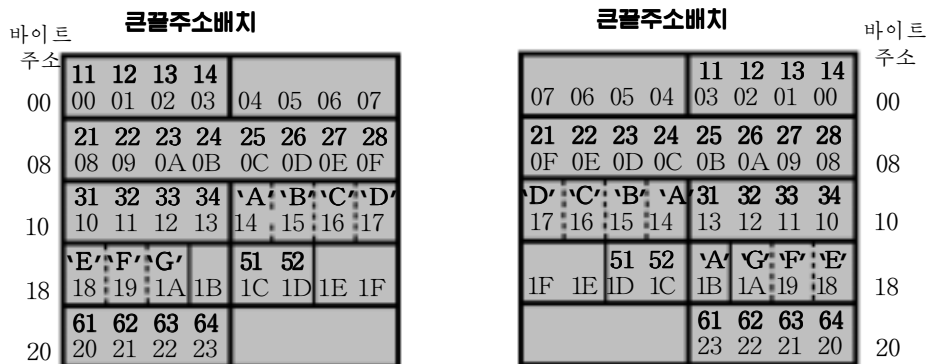



그림 9-18. C 자료구조와 그의 끌배치도의 실례[IBM 94]

그림 9-18 은 끌배치성이 어떻게 주소지정과 바이트순서를 결정하는가를 보여 주고 있다. 그림에서 꼭대기에 있는 C 구조는 여러가지 자료형을 가지고 있다. 그림에서 낮은 부분의 왼쪽에 있는 기억기배치도는 큰끌콤퓨터인 경우 자료구조의 번역결과이며 오른쪽 아래에 있는 배치도는 작은끌콤퓨터인 경우의 자료구조번역결과이다. 모든 경우에 기억기는 64bit 행의 렬로 된다. 큰끌인 경우 기억기는 왼쪽에서 오른쪽으로, 우에서 아래로 전개되며 작은끌인 경우 오른쪽에서 왼쪽으로, 우에서 아래로 전개된다. 이 배치도들은 규칙이 없이 제멋대로라는것을 강조한다. 한행안에서 왼쪽에서 오른쪽으로 혹은 오른쪽에서 왼쪽으로의 두가지 양식을 모두 리용할수 있다. 이것은 어디까지나 기억기할당과는 무관계한 묘사와 관련한 문제이다. 이와 같은 자료구조에 대한 몇가지 조사결과들은 다음과 같다.

- 매 자료항목은 두 양식에서 같은 주소를 가진다. 실례로 16 진값 2122232425262728 의 배단어주소는 08 이다.
- 임의의 주어 진 여러바이트스칼라값에 대한 작은끌구조의 바이트순서화는 큰끌배치구조의 바이트순서화와 반대이다.
- 끌배치성은 한 구조내에서의 자료항목의 순서화에 영향을 주지 않는다. 결국 4 개의 문자단어 c 는 바이트역전을 나타내지만 7 개의 문자바이트배렬 d 는 바이트역전을 나타내지 않는다. 그러므로 두 구조에서 d 의 매 개별요소들의 주소는 같다.

바이트가 수직으로 배렬된 그림 9-19 에 보여 준 기억기구조를 고찰하면 끌배치성의 결과를 보다 더 명백히 알수 있다. 끌배치성에서 어느 방식이 우월한가에 대한 견해는 일반적으로 일치하지 않는다.

그러나 다음과 같은 점에서 큰끌방식이 더 우월하다고 볼수 있다.

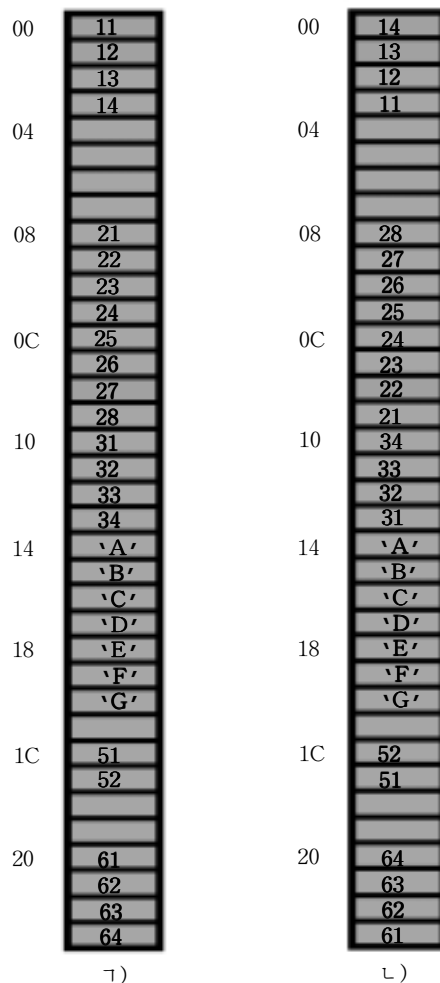



그림 9-19. 그림 9-18 의 다른 한가지 고찰
ㄱ-큰끌배치, ㄴ-작은끌배치

- **문자렬분류:** 큰끌처리장치는 옹근수정렬문자렬비교에서 작은끌방식보다 고속이다. 옹근수 ALU는 병렬로 여러바이트를 비교할수 있다.
- **10진/ASCII 인쇄:** 모든 값들을 혼동없이 왼쪽에서 오른쪽으로 인쇄할수 있다.
- **일치한 순서:** 큰끌처리장치는 옹근수와 문자렬을 같은 순위로 기억한다(맨 웃자리바이트가 먼저 온다.).

또한 다음과 같은 점에서는 작은끌방식이 더 우월하다.

- 큰끌처리장치는 32bit 옹근수주소를 16bit 옹근수주소로 바꾸는 경우 맨 아래자리바이트를 리용하여 더하기를 해야 한다.
- 작은끌격식으로는 보다 더 높은 정확도의 연산을 수행하기가 쉽다. 즉 맨 아래자리 바이트를 찾아서 역방향으로 옮기지 않아도 된다.

이와 같은 차이는 그닥 중요하지 않으며 보다 더 중요한것은 끌배치격식의 선택이 그 이전 기계들에 맞겠는가 하는것이다.

PowerPC는 큰 및 작은끌방식을 모두 지원하는 쌍끌처리장치이다. 쌍끌기본방식은 쏘프트웨어개발자들이 다른 기계로부터 조작체계나 응용쏘프트들을 옮길 때 임의의 방식을 선정해도 되게 해준다. 끌배치방식은 조작체계가 선정하며 그에 따라 처리가 진행된다. 방식이 선택되자마자 뒤따르는 모든 기억기넣기 및 기억은 그 방식의 기억기주소지정모형에 의해 규정된다. 이 하드웨어특징을 지원하기 위하여 2개의 비트가 처리상태의 한 부분으로서 조작체계에 의해 유지되는 기계상태등록기(MSR)에 보존된다. 한비트는 조작체계핵심부가 실행하는 끌배치방식을 규정하며 다른 한비트는 처리장치의 현재 동작방식을 규정한다.

2. 비트순서화

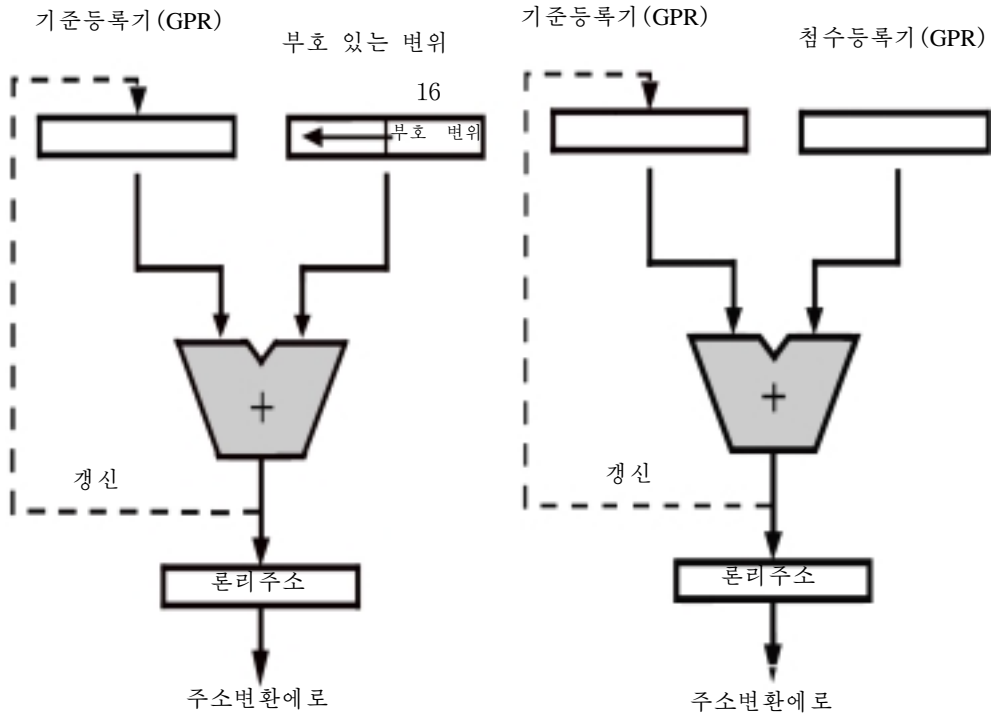
한바이트안의 비트를 순서화하는데서는 다음의 두가지 문제에 직면하게 된다.

1. 첫 비트를 비트 0 혹은 1로 보는가?
2. 맨 아래자리비트를 바이트의 가장 작은 무게를 가진 비트(작은끌) 혹은 바이트의 가장 큰 무게를 가진 비트(큰끌)로 할당하는가?

이 질문에 대하여서는 모든 기계들에서 꼭 같은 방법으로 대답할수 없다. 일부 콤퓨터들에서는 각이한 상황에 따라서 이 대답이 서로 다르다. 더우기 이것은 한바이트안에서의 큰 혹은 작은끌비트순서화와 일치하지 않는다. 프로그람작성자는 개별적인 비트들을 조작하는 경우에 이 문제를 반드시 고려해야 한다.

한가지 더 관심해야 할것은 자료가 비트렬로 직렬로 전송될 때이다. 이때 개별적인 바이트가 전송된다면 제일 무게가 큰 비트를 먼저 전송하는가, 제일 무게가 작은 비트를 먼저 전송하는가 하는 문제가 생긴다. 설계가는 들어 오는 비트들이 알맞게 조종되는가를 확인해야 한다. 이와 같은 문제는 [JAME90]참고문헌을 보면 더 잘 알수 있다.

제 10 장. 명령모임: 주소화방식과 형식

◆◆◆

◆ 명령에서 연산수참조는 연산수의 실지값(직접값)이나 연산수주소에 대한 참조를 포함하고 있다. 여러가지 주소화방식들이 각이한 명령모임들에 리용되고 있다. 여기에는 직접 (연산수주소가 주소마당안에 있다.), 간접 (주소마당이 연산수주소를 가지는 위치를 가리킨다.), 등록기, 등록기간접, 등록기값이 연산수주소를 만들도록 주소값에 더해 지는 여러가지 형의 변위 등 주소화방식들이 있다.

◆ 명령형식은 명령에서의 지면배치마당을 정의한다. 명령형식설계는 고정 혹은 가변길이의 명령길이, 조작코드와 매 연산수참조에 할당된 비트수, 주소화방식의 결정방법 등을 비롯하여 복잡하다.

◆◆◆

제 9 장에서는 명령모임이 무엇을 하는가를 기본으로 고찰하였다. 특히 연산수의 종류와 기계명령에 의해 규정되는 연산들을 정의하였다. 이 장에서는 명령의 연산수와 연산을 규정하는 방법을 기본으로 고찰한다. 우선 연산수의 주소가 어떻게 규정되는가를 보고 그다음 명령비트들이 어떻게 구성되며 명령의 연산수주소와 연산을 어떻게 정의하는가를 고찰한다.

제1절. 주소화방식

일반적으로 명령형식에서 주소마당은 상대적으로 작다. 콤퓨터에서는 큰 범위의 주기억기(일부 체계들에서는 가상기억기)위치를 참조하기 위하여 여러가지 주소화기술을 리용하고 있다. 이 주소화방식들의 선정에서는 주소범위와 주소화의 유연성, 기억기참조수와 주소계산에서의 복잡성사이 관계를 잘 알고 합리적인것을 선택해야 한다. 이 절에서는 다음과 같은 가장 일반적인 주소화방식들에 대하여 고찰한다.

- 직접값
- 직접
- 간접
- 등록기
- 등록기간접
- 변위
- 탄창

이 방식들을 그림 10-1에 보여 주었다. 이 절에서는 다음과 같은 표기를 리용한다.

A = 명령에서 주소마당의 내용
R = 등록기를 리용하는 명령에서 주소마당의 내용
EA = 참조한 연산수를 가지는 위치의 유효주소
(X) = 위치 X의 내용

표 10-1은 매 주소화방식에서 수행되는 주소계산을 보여 주고 있다.

구체적인 론의를 하기에 앞서 다음과 같은 두가지 문제를 간단히 고찰한다. 우선 가상적으로 모든 콤퓨터구성방식들은 우에서 언급한 주소화방식들가운데서 하나이상의 주소화방식을 제공한다. 이때에는 조종장치가 개별적인 명령들에서 어느 주소방식을 리용하고 있는가를 어떻게 알아 내는가 하는 문제가 생긴다.

이를 위한 여러가지 방법들이 있다. 보통 여러가지 주소화방식에 각이한 조작코드를 대응시키는 방법, 명령에서 하나 혹은 그이상의 비트들을 **방식마당**으로 리용하는 방법 등이 있다.

다음으로 **유효주소**(*EA*)의 해신에 관한 문제이다. 가상기억기가 없는 체계에서 유효주소는 주기억기주소 혹은 등록기로 된다.이 주소를 실지 물리주소로 바꾸는것은 페지화기구의 기능이며 프로그람작성자에게는 보이지 않는다.

표 10-1. 기본주소화방식

방식	알고리듬	기본우점	기본부족점
직접값	연산수 = A	기억기참조가 없다.	제한된 연산수크기
직접	유효주소 = A	단순하다.	제한된 주소공간
간접	유효주소 = (A)	큰 주소공간	여러 기억기참조
등록기	유효주소 = R	기억기참조가 없다.	제한된 주소공간
등록기간접	유효주소 = (R)	큰 주소공간	외부기억기참조
변위	유효주소 = A+(R)	유연성이 있다.	복잡성
탄창	유효주소 = 탄창의 정점	기억기참조가 없다.	제한된 능력

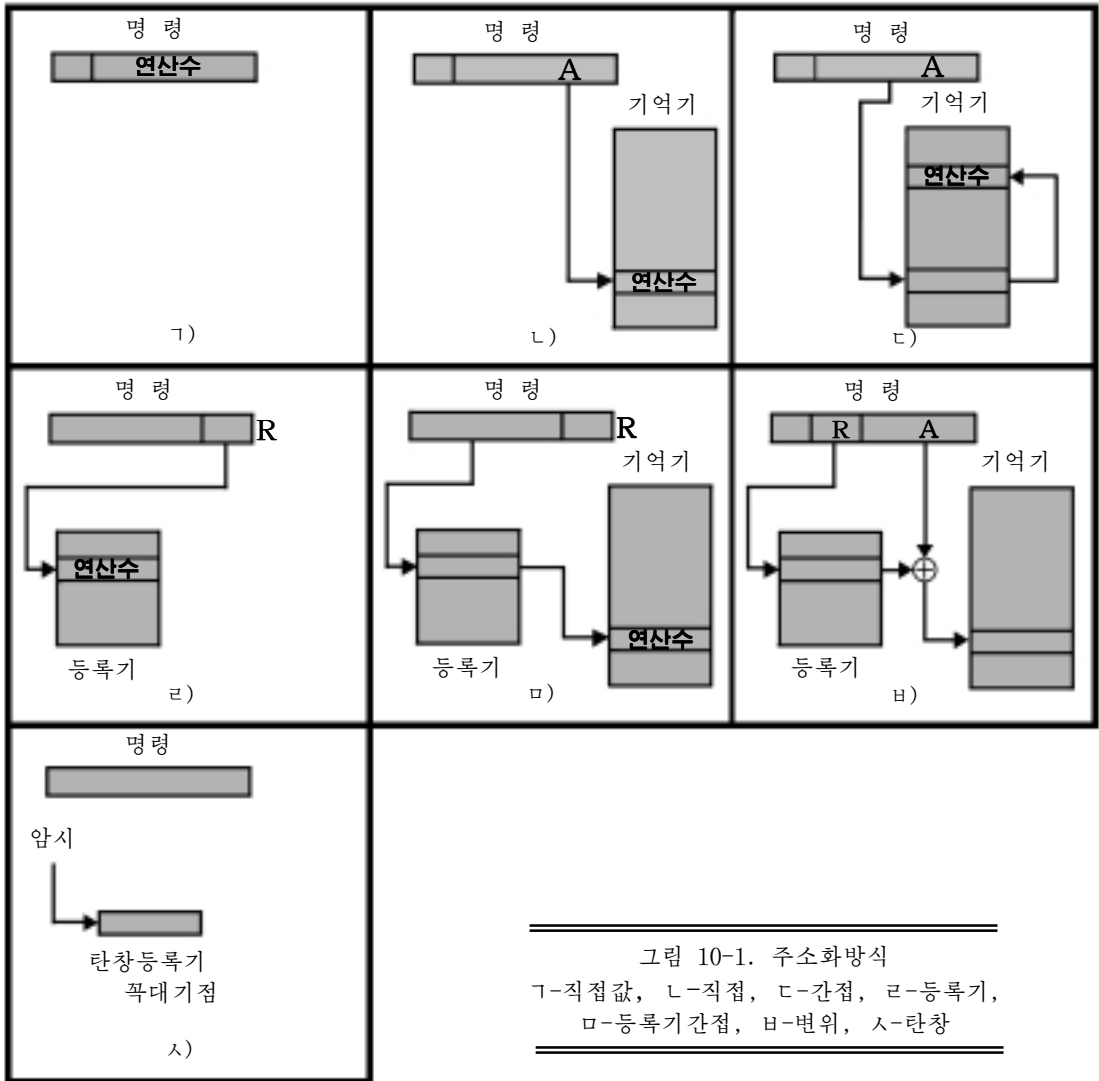



그림 10-1. 주소화방식
ㄱ-직접값, ㄴ-직접, ㄷ-간접, ㄹ-등록기,
ㅁ-등록기간접, ㅂ-변위, ㅅ-탄창

1. 직접값주소화

가장 간단한 양식의 주소화방식은 연산수가 명령내에 실지로 존재하는 직접값주소화방식이다.

연산수 ＝ A

이 방식은 상수를 정의하여 리용하거나 변수들의 값을 초기화하는데 리용할수 있다. 수는 2 의 보수양식으로 기억되며 연산수마당의 제일 왼쪽에 있는 비트가 부호비트로 리용된다. 연산수가 자료등록기에 넣어 지면 부호비트는 완전자료단어크기로 확장된다.

직접값주소화의 우점은 명령꺼내기외에 연산수를 얻는데 기억기참조를 리용하지 않는다는것이다. 따라서 명령주기에서 하나의 기억기나 캐쉬주기를 절약하게 된다. 이 방

식의 부족점은 수의 크기가 주소마당의 크기로 제한된다는것이다. 이것은 대부분의 명령모임들에서 주소마당이 단어길이에 비해 작기때문이다.

2. 직접주소화

이 방식은 매우 단순한 주소화방식으로서 주소마당이 연산수의 유효주소를 포함한다.

유효주소= A

이 기술은 초기에 나온 콤퓨터들에서 일반적으로 리용되였으며 지금도 많은 콤퓨터체계에 적용되고 있다.이 방식은 하나의 기억기참조만을 요구할뿐 특별한 계산은 요구하지 않는다. 이 방식의 명백한 제한성은 제한된 주소만을 리용할수 있다는것이다.

3. 간접주소화

직접주소화방식에서는 보통 주소마당의 길이가 단어길이보다 더 작으며 따라서 주소범위가 제한된다. 이 문제를 해결하기 위하여 주소마당을 기억기에서의 단어주소로 되게 하여 차례로 완전한 길이의 연산수주소를 가지게 한다. 이와 같이 주소를 지정하는 방식이 **간접주소화방식**이다.

유효주소=（A）

이미 앞에서 언급한바와 같이 웃식에서 괄호는 무엇의 내용을 의미하는것으로 해석된다. 이 방법의 우점은 *N*bit 의 단어길이인 경우 리용할수 있는 주소공간이 2^N 이라는것이다. 부족점은 명령실행이 연산수를 불러 내는데 두번의 기억기참조를 필요로 한다는것이다. 즉 하나는 주소를 얻는것이고 다른 하나는 그의 값을 얻는것이다.

주소화에 쓸수 있는 단어들의 수가 2^N 이지만 한번에 참조할수 있는 서로 다른 유효주소의 수는 2^K 으로 제한된다. 여기서 K 는 주소마당의 길이이다. 가상기억기환경에서 모든 유효주소위치는 임의의 처리의 페지 0 으로 제한될수 있다. 한 명령의 주소마당이 작으므로 이것은 페지 0 에서 존재하는 작은 수의 직접주소를 산생한다（제한성은 단지 페지크기가 2^K 과 같거나 그보다 더 커야 한다는것이다.）. 처리가 가동하면 실지기억기에 주소마당이 남도록 페지 0 에 대한 참조가 반복된다. 따라서 간접기억기참조는 많아서 2페지가 아니라 암시적으로 한페지를 참조한다.

드물게 리용되기는 하지만 간접주소화에서는 여러 수준 혹은 종속련결된 간접주소화방식을 리용한다.

유효주소 =（...（A)...）

이 경우에 전체-단어주소의 한비트가 간접기발(I)이다. 1bit가 0이면 단어는 유효주소를 포함하며 1bit가 1이면 간접적인 다른 수준이 동반된다. 이 방법에는 특별한 우점은 없으며 부족점으로는 연산수를 불러 내는데 필요한 기억기참조가 셋 혹은 그이상이라는것이다.

4. 등록기주소화

등록기주소화는 직접주소화와 류사하다. 단지 차이점은 주소마당이 주기억기주소가 아니라 등록기라는것이다.

유효주소= R

등록기들을 참조하면 주소마당은 3 혹은 4bit 로 되며 따라서 이 주소마당을 리용하면 8

혹은 16개의 일반목적등록기를 참조할수 있다.

등록기주소화의 우점은 명령에서 작은 주소마당을 요구하며 기억기참조가 필요 없다는것이다. 제 4 장에서 이미 고찰한바와 같이 CPU 내에 있는 등록기를 리용한 기억기호출시간은 주기억기주소를 리용한 호출시간에 비해 훨씬 작다. 등록기주소화의 부족점은 주소공간이 대단히 제한된다는것이다.

등록기주소화방식이 명령모임에서 자주 리용된다면 그것은 CPU 등록기들이 많은 부담을 받는다는것을 말해 준다. 주기억기위치에 비해 등록기는 그수가 제한되여 있으므로 그것들이 효과적으로 리용되는 경우에만 은을 낼수 있다. 모든 연산수를 주기억기로부터 등록기로 옮기면 연산을 즉시에 수행하고 주기억기로 돌아 오게 되며 이때 불필요한 중간계단이 추가된다. 그대신 등록기연산수를 다중연산에 리용한다면 그때에는 실지 보관을 하게 된다. 그 실례가 바로 계산에서의 중간결과이다. 이제 2 의 곱하기알고리듬을 쏘프트웨어적으로 실현한다고 하자. 그러면 그림 8-12의 흐름도에서 A라는 표식이 붙은 위치가 여러번 참조되며 이것은 주기억기위치가 아니라 등록기로 실현된다.

값을 등록기에 남게 하겠는가, 주기억기에 기억하겠는가 하는 문제는 어디까지나 프로그람작성자의 권한에 속한다. 오늘날 대다수의 CPU 들은 아쎔블리언어프로그람작성자들이 효과적인 집행프로그람을 작성할수 있도록 여러개의 일반목적등록기들을 가지고 있다.

5. 등록기간접주소화

등록기주소화가 직접주소화와 류사한 방식이라면 등록기간접주소화는 간접주소화와 류사한 방식이다. 단지 차이점은 주소마당이 기억기위치 아니면 등록기에 귀착되는가 하는데 있다. 따라서 등록기간접주소화방식은 다음과 같이 표시할수 있다.

유효주소＝(R)

등록기간접주소화의 우점과 제한성은 기본적으로 간접주소화와 같다. 이 두 주소화방식에서 주소마당의 주소공간의 제한성은 그 마당을 주소를 가지는 단어-길이위치로 되게 하면 해소될수 있다. 등록기간접주소화는 간접주소화방식보다 기억기참조를 하나 적게 한다.

6. 변위주소화

매우 강력한 주소화방식은 직접주소화와 등록기간접주소화의 능력을 결합한것이다. 이것은 그 리용상황에 따라 각이하게 이름이 불리워 지고 있지만 그 기본기구는 같다. 여기서는 **변위주소화**라고 한다.

유효주소 ＝A＋(R)

변위주소화는 모든 명령이 2개의 주소마당을 가진다. 그중에서 적어도 하나는 명시적이여야 한다. 주소마당에 있는 값(값＝A)은 직접 리용된다. 다른 주소마당 혹은 조작코드에 기초한 암시적인 참조는 그의 내용이 유효주소를 만들도록 A에 더해 지는 등록기에 귀착된다.

변위주소화방식의 가장 일반적인 리용에는 다음의 세가지가 있다.

- 상대주소화
- 기준-등록기주소화
- 첨수주소화

상대주소화

상대주조지정에서 암시적으로 참조되는 등록기는 프로그람계수기이다. 즉 현재명령

주소는 유효주소를 만들도록 주소마당에 더해 진다. 주소마당은 이 연산을 위하여 2 의 보수로 취급된다. 결국 유효주소는 명령주소의 상대적인 변위이다.

상대주소화는 제4장과 제7장에서 고찰한 국부성의 개념을 리용한다. 대부분 기억기 참조가 실행중인 명령과 상대적으로 가까이에 있는 경우에 상대주소화방식을 리용하면 명령에서 주소비트를 줄일수 있다.

기준-등록기주소화

기준등록기주소화방식에 대한 설명은 다음과 같다. 참조된 등록기는 기억기주소를 가지며 주소마당은 그 주소로부터의 변위(보통 부호 없는 옹근수표현)를 가진다. 등록기참조는 명시적이거나 암시적일수 있다.

기준등록기주소화는 또한 국부적인 기억기참조를 리용한다. 이것은 제 7 장에서 고찰한 토막화를 실현하는 편리한 수단이다. 이 주소화방식에서는 일부 단일토막등록기가 채용되여 암시적으로 리용된다. 이때 프로그람작성자는 토막의 기준주소를 가지는 등록기를 선정하고 명령으로 그것을 명시적으로 참조해야 한다. 후자의 경우에는 주소마당의 길이가 K 이고 가능한 등록기의 수가 N 이면 하나의 명령으로 N 개의 2^k 단어중 임의의 하나만을 참조할수 있다.

첨수주소화

이 방식에 대한 설명은 다음과 같다. 주소마당은 주기억기주소를 참조하며 참조된 등록기는 그 주소로부터의 정방향변위를 가진다. 이것은 기준등록기주소화방식과 반대라는것을 강조한다. 첨수주소화에서는 주소마당이 기억기주소이므로 일반적으로 기준등록기명령에 비해 더 많은 비트의 주소마당을 가지게 된다. 또한 기준등록기주소화에서 쓸모 없었던것을 첨수주소화에서는 일부 쓸모 있게 만든것도 있다. 그럼에도 불구하고 유효주소를 계산하는 방법은 기준등록기주소화와 첨수주소화에서 같으며 두 경우에 등록기참조는 때로는 명시적이고 때로는 암시적이다(CPU 형태에 따라서).

첨수주소화방식은 반복연산을 수행하는데 효과적으로 리용한다. 실례로 위치 A 로부터 시작하여 기억된 수들의 목록을 생각해 보자. 이제 이 목록의 매 요소에 1 을 더한다고 하자. 이때에는 우선 매값을 불러 내야 하며 그다음 거기에 1 을 더하고 결과를 다시 기억해야 한다. 필요한 유효주소의 렬은 목록의 마지막위치까지의 A, A + 1, A + 2, …이다. 첨수주소화를 리용하면 이것을 쉽게 할수 있다. 즉 값 A를 명령의 주소마당에 기억하고 첨수등록기로 선정된 등록기를 0 으로 초기화한다. 매번 연산이 진행된후에는 첨수등록기가 하나씩 증가된다.

이와 같이 첨수등록기들은 일반적으로 반복과제를 수행하는데 리용되므로 매번 그것을 참조한후에는 그 내용을 증가 혹은 감소시켜야 한다. 이로부터 일부 체계에서는 이것을 같은 명령주기의 한 부분으로 자동적으로 수행하는데 이것을 보통 자동첨수주소화라고 한다. 만일 어떤 등록기들이 첨수주소화에만 리용된다면 그때 자동첨수주소화가 암시적으로, 자동적으로 진행될것이다. 일반목적등록기들이 리용된다면 자동첨수연산은 명령내의 한비트로 이것을 신호해야 한다. 증가형자동첨수주소화는 다음과 같은 식으로 표시할수 있다.

$$(EA) = A + (R)$$
$$R \leftarrow (R) + 1$$

일부 콤퓨터에서는 간접주소화와 첨수주소화를 모두 제공하며 같은 명령으로 이 두 방식을 리용할수 있게 되여 있다. 여기에는 두가지 가능성이 있다. 즉 첨수주소화를 간접주소화의 전 혹은 후에 수행하는것이다.

첨수주소화를 간접주소화후에 수행하면 **뒤첨수주소화**（postindexing）라고 한다.

EA = （A)＋(R)

우선 주소마당의 내용을 직접주소를 가지고 있는 기억기위치에 호출하도록 리용한다. 그 다음 이 주소가 등록기값으로 된다. 이 기술은 고정된 형식의 많은 자료블로크들중 어느 하나를 호출하는데 효과적으로 리용할수 있다. 실례로 이미 제 7 장에서 조작체계는 모든 처리를 위하여 처리조종블로크를 리용하여야 한다는것을 서술하였다. 이때 수행되는 연산들은 블로크를 조작하는것과는 상관없이 모두 같다. 따라서 블로크를 참조하는 명령의 주소는 변수지적자가 처리조종블로크의 시작위치（값＝ A）로 된다. 첨수등록기는 블로크내에서의 변위를 가진다.

첨수주소화가 간접주소화보다 먼저 수행되면 **앞첨수주소화**(preindexing)이라고 한다.

EA = （A ＋(R))

주소는 단순히 첨수주소화방식으로 계산된다. 그러나 이 경우에는 계산된 주소가 연산수가 아니라 연산수의 주소이다. 이 기술을 리용한 실례는 여러궤도분기표를 만드는것이다. 프로그람의 특별한 지점에서 조건에 따라서 여러 위치들중 어느 하나로 갈라 진다. 주소표는 위치 A 로부터 시작하게 설정된다. 이 표에 첨수를 붙이면 필요한 위치를 찾을수 있다.

보통 명령모임은 앞 및 뒤첨수주소화방식을 가지고 있지 않다.

7. 탄창주소화

탄창은 부록 9-1 에서 정의한바와 같이 위치들의 선형적인 배렬이다. 탄창은 때때로 아래로 **밀어넣기목록** 혹은 **후입선출대기렬**이라고도 한다. 탄창은 예약된 위치들의 블로크이다. 항목들은 임의의 주어 진 시간에 블로크를 부분적으로 채우도록 탄창의 꼭대기에 덧붙는다. 탄창에서 중요한것은 탄창지시기로서 그의 값은 탄창의 꼭대기주소이다. 탄창의 꼭대기 두개의 요소들은 CPU 등록기에 있을수도 있는데 이 경우에 탄창지시기는 탄창의 제 3 의 요소를 참조한다（그림 9–14 ㄱ）. 탄창지시기는 등록기에 보존되며 따라서 기억기에서 탄창위치에 대한 참조는 사실상 등록기간접주소이다.

탄창주소화방식은 암시적인 주소화방식의 한 형태이다. 기계명령은 기억기참조를 포함하지 않지만 암시적으로 탄창의 꼭대기에서 동작한다. 탄창은 전통적으로 일반성을 띠고 리용되지 않지만 극소형처리장치에서는 거의나 보편적으로 리용되고 있다.

제 2 절. Pentium Ⅱ와 PowerPC 의 주소화방식

1. Pentium Ⅱ의 주소화방식

이미 앞에서 본 그림 7-22 를 다시 고찰해 보자. 여기서는 Pentium Ⅱ주소번역기구가 토막의 변위인 가상 혹은 유효주소라고 하는 주소를 만들어 내고 있다. 토막의 시작주소와 유효주소의 합은 선형주소를 이룬다. 페지화가 리용되고 있는 경우에 이 선형주소는 물리주소가 되도록 페지번역기구를 거쳐야 한다. 뒤에서의 론의에서는 이 마지막단계를 무시하는데 그것은 이것이 명령모임에서나 프로그람작성자에게 명백하기때문이다.

Pentium Ⅱ는 고급언어프로그람을 효과적으로 실행할수 있도록 여러가지 주소화방식을 가지고 있다. 그림 10–2 에는 주소화방식에 리용되는 하드웨어를 보여 주었다. 참조대상인 토막은 토막등록기에 의해 정해 진다. Pentium Ⅱ에는 6 개의 토막등록기가 있다. 이 토막등록기들은 실행의 전후관계와 명령에 따라 알맞게 참조된다. 모든 토막등록기는

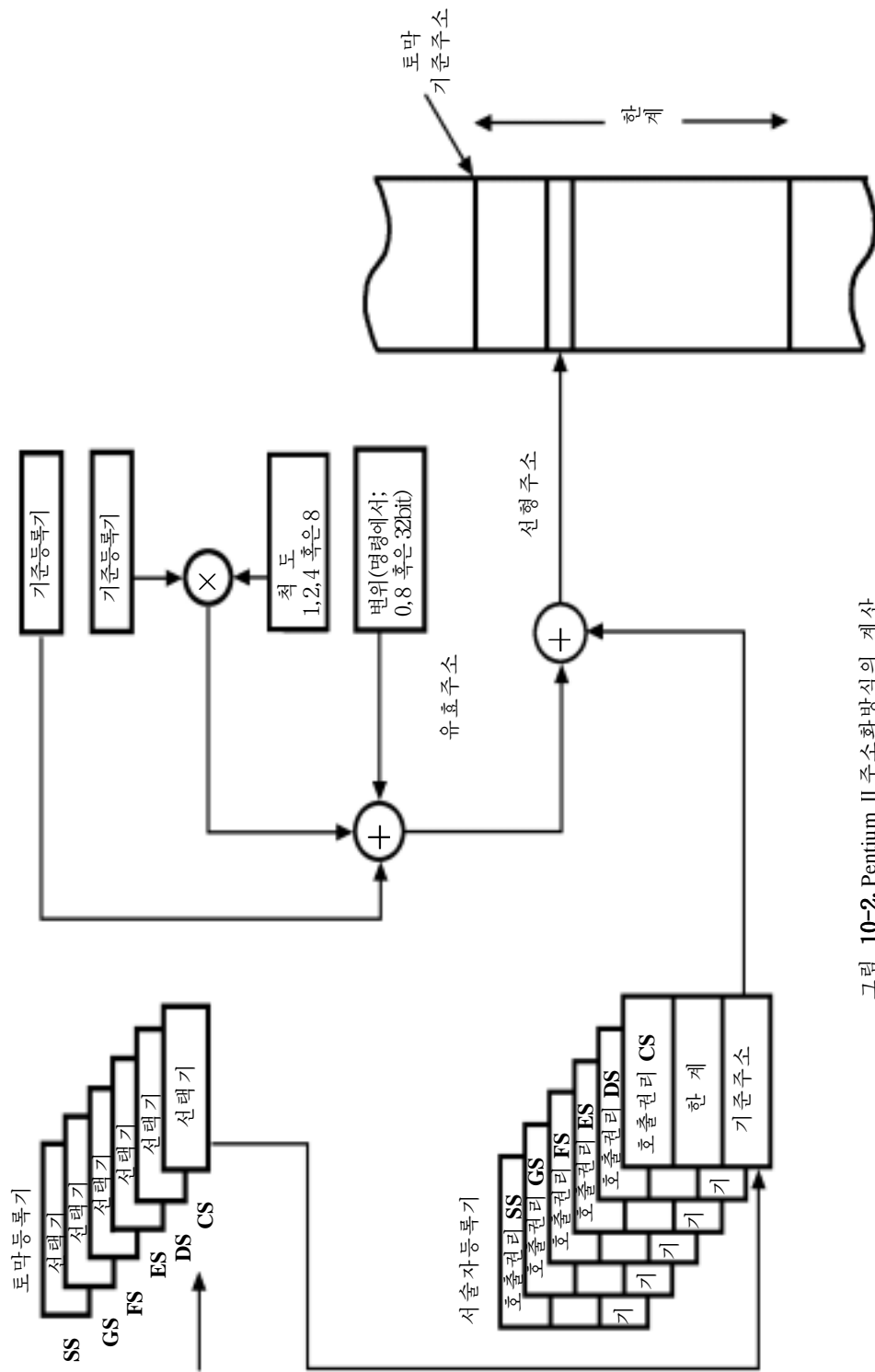



그림 10-2. Pentium II 주소화방식의 계산

프로그람작성자가 볼수 없는 토막서술자등록기와 관계를 가지는데 이 토막서술자등록기는 토막의 시작주소는 물론 한계(길이)와 같은 토막에 대한 호출정보를 가지고 있다. 이외에도 주소를 만드는데 리용하는 기준등록기와 첨수등록기가 있다.

표 10-2 는 12 개의 Pentium Ⅱ 주소화방식을 보여 주고 있다. 이제부터 이 주소화방식들을 차례로 고찰해 보자.

표 10-2. Pentium Ⅱ의 주소화방식

방식	알고리듬
직접값	연산수 = R
등록기연산수	LA = R
변위	LA = (SR) + A
기준	LA = (SR) + (B)
변위를 가진 기준	LA = (SR) + (B) + A
변위를 가진 척도첨수	LA = (SR) + (I)*S + A
첨수와 변위를 가진 기준	LA = (SR) + (B) + (I) + A
척도첨수와 변위를 가진 기준	LA = (SR) + (I)*S + (B) + A
상대	LA = (PC) + A
LA – 선형주소	R – 등록기
(X)– X의 내용	B – 기준등록기
SR – 토막등록기	I – 첨수등록기
PC – 프로그람계수기	S – 척도인자
A – 명령에서 주소마당의 내용	

직접값방식은 연산수를 명령안에 포함시킨다. 이 연산수는 바이트, 단어 혹은 배단어자료일수 있다.

등록기연산수방식에서는 연산수가 등록기에 위치한다. 자료전송, 산수-론리연산명령과 같은 일반명령에서는 연산수로 32bit 의 일반등록기(EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP)들중 어느 하나, 16bit의 일반등록기(AX, BX, CX, DX, SI, DI,SP, BP)들중 어느 하나, 8bit의 일반등록기(AH, BH, CH, DH, AL, BL, CL, DL)들중 어느 하나를 리용한다. 류점수연산에서 64bit 연산수로는 두개의 32bit 등록기쌍을 리용한다. 또한 일부 명령들에서는 토막등록기(CS, DS, ES, SS, FS, GS)를 참조한다.

나머지주소화방식들은 기억기위치를 참조한다. 기억기위치는 토막의 위치와 토막의 시작으로부터의 변위를 가지도록 규정되여야 한다. 어떤 경우에는 토막이 명시적으로 규정되며 또 어떤 경우에는 토막을 암시적으로 할당하는 단순한 규칙에 의해 규정되기도 한다.

변위방식에서 연산수의 변위 (그림 10-2 의 유효주소) 는 8, 16 혹은 32bit의 변위로서 명령의 한 부분으로 된다. 토막화에서는 명령내의 모든 주소가 순수 한 토막내에서의 변위를 가리킨다. 변위주소화방식은 일부 콤퓨터들에서만 찾아 볼수 있는데 그것은 이미 언급한바와 같이 이 방식이 명령의 길이가 길기때문이다. Pentium Ⅱ인 경우에 변위값은 32bit정도로 되며 따라서 6byte의 명령을 만든다. 변위주소화방식은 전역변수를 참조하는데 효과적으로 쓸수 있다.

나머지주소화방식은 명령의 주소부분이 처리장치로 하여금 주소를 찾아 보도록 하는 간접방식들이다. **기준방식**에서는 8, 16 혹은 32bit의 등록기들중 어느 하나가 유효주소를 가진다. 이 방식은 등록기간접주소화방식과 같다.

변위를 가진 기준방식에서는 명령이 기준등록기에 더해 지는 변위를 포함하며 기준등록기로는 일반목적등록기들중에서 임의의것을 쓴다. 이 방식의 리용을 실례로 들어 고찰하면 다음과 같다.

- 국부변수령역의 시작위치를 지시하는 콤파일러에 리용된다. 실례로 기준등록기는 대응하는 수속에서 리용하는 국부변수들이 들어 있는 탄창의 시작위치를 지적한다.
- 요소크기가 2, 4 혹은 8bit 가 아닌 등록기들에는 배렬내에서의 첨수지정에 리용하며 첨수등록기를 첨수지정에 리용할수 없다. 이 경우에 변위는 배렬의 시작을 지시하며 기준등록기에는 배렬내에 있는 특별한 요소에 대한 변위를 결정하는 계산결과가 등록된다.
- 레코드마당을 호출하는데 리용된다. 기준등록기는 레코드의 시작을 지시하며 변위는 이 마당에 대한 편위이다.

변위를 가진 척도첨수방식에서는 명령이 첨수등록기에 더해 지는 변위를 포함하고 있다. 첨수등록기로는 ESP 라고 하는 등록기를 제외한 임의의 일반목적등록기를 리용할수 있다. ESP 는 일반적으로 탄창처리에 리용된다. 유효주소는 첨수등록기의 내용을 척도인자 1, 2, 4 혹은 8 에 곱하고 그 결과에 변위를 더하여 얻는다. 이 방식은 배렬의 첨수화에 매우 편리한 방식이다. 척도인자 2 는 16bit 의 옹근수배렬에 리용되며 척도인자 4 는 32bit 의 옹근수 혹은 류점수배렬에, 척도인자 8 은 배정확도류점수배렬에 리용된다.

첨수와 변위를 가진 기준방식에서는 기준등록기와 첨수등록기의 내용들과 유효주소를 이루는 변위를 모두 더한다. 기준등록기로는 임의의 일반목적등록기를 리용하며 첨수등록기로는 ESP 를 제외한 임의의 일반목적등록기를 리용한다. 이 주소화방식은 탄창틀에 대한 국부배렬을 호출하는데 리용할수 있다. 이 방식은 또한 2 차원배렬을 지원하는데도 리용할수 있다. 이 경우에 변위는 배렬의 시작을 지시하며 매 등록기는 1 차원배렬을 조종한다.

변위를 가진 기준척도첨수방식에서는 척도인자를 곱한 첨수등록기의 내용과 기준등록기의 내용, 변위를 모두 더한다. 이 방식은 배렬이 탄창틀에 기억되여 있는 경우 효과적이다. 이 경우에는 배렬요소들의 길이가 2, 4 혹은 8byte 로 된다. 이 방식은 또한 배렬요소가 2, 4 혹은 8byte 의 길이를 가지는 경우 2 차원배렬의 첨수지정에 효과적으로 리용할수 있다.

상대주소화방식은 조종흐름명령에 리용된다. 변위는 다음명령을 가리키는 프로그람계수기의 값에 더해 진다. 이 경우에는 변위가 부호 있는 바이트, 단어 혹은 배단어값으로 취급되며 이 값은 프로그람계수기의 주소를 증가시키거나 감소시킨다.

2. PowerPC 주소화방식

대다수 RISC 콤퓨터들에서와 마찬가지로 PowerPC 는 Pentium Ⅱ와 대다수 CISC 콤퓨터들과는 달리 단순하면서도 상대적으로 간단한 주소화방식을 리용한다. 표 10-3 에서 보여 준바와 같이 이 방식들은 명령의 형에 따라서 편리하게 분류되여 있다.

넣기/기억주소화

PowerPC 는 두 대치되는 주소화방식인 넣기 및 기억주소화방식을 가지고 있다(그림 10-3). 간접주소화인 경우 명령은 기준등록기에 더해 지는 16bit 의 변위를 포함하는데 이때 기준등록기로는 임의의 일반목적등록기를 리용할수 있다. 반대로 명령은 새로 계산된 유효주소를 기준등록기에 넣어 그의 현재내용을 갱신하도록 한다. 이 갱신선택은 고

리에서 배렬의 첨수지정에 효과적으로 리용할수 있다.

표 10-3. PowerPC 의 주소화방식

방식	알고리듬
	넣기/기억주소화
간접	EA = (BK) + D
간접첨수	EA = (BR) + (IR)
	분기주소화
절대	EA = I
상대	EA = (PC) + I
간접	EA = (L / CR)
	고점수계산
등록기	EA = GPR
직접값	연산수 = I
	류점수계산
등록기	EA = FPR

EA	–	유효주소	(X)	–	X 의 내용
BR	–	기준등록기	IR	–	첨수등록기
L/CK	–	련결 혹은 계수등록기	GPR	–	일반목적등록기
FPR	–	류점수등록기	D	–	변위
I	–	직접값	PC	–	프로그람계수기

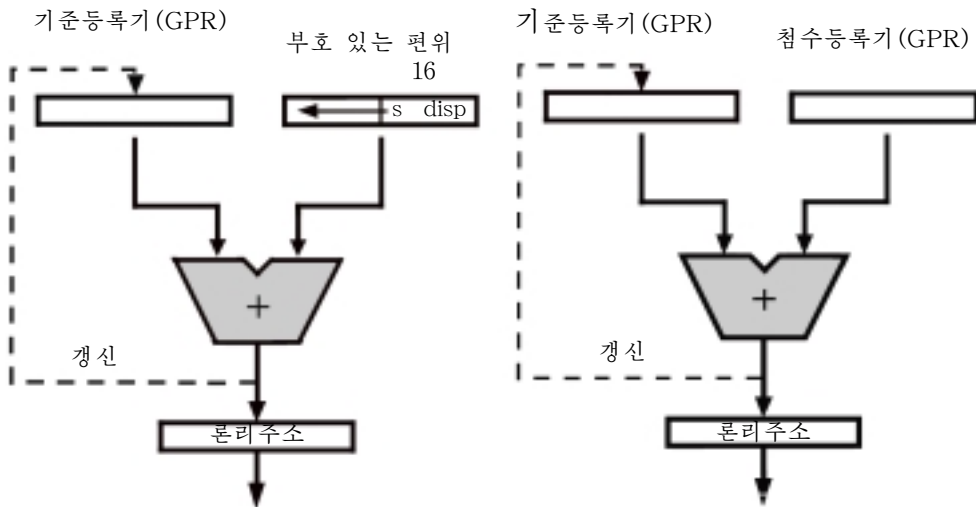



그림 10-3. Power PC 기억기연산수주소지정방식[DIEF94b]

넣기 및 기억명령을 리용하는 다른 하나의 주소화방식은 **간접첨수지정방식**이다. 이 방식에서는 명령이 기준등록기와 첨수등록기를 참조한다. 이때 두 등록기들로는 임의의 일반목적등록기를 리용할수 있다. 유효주소는 두 등록기의 내용을 더하여 얻어 진다. 결국 기준등록기는 새로운 유효주소로 갱신된다.

분기주소화

분기주소화방식에는 세가지가 있다. **절대주소화방식**을 무조건분기명령에 리용하는 경우 다음명령의 유효주소는 명령내에 있는 24bit 의 직접값으로부터 얻어 진다. 24bit 의 값은 맨 아래자리비트의 다음에 두개의 0 을 첨가하고 부호를 확장하면 32bit 의 값으로 된다. 이것은 모든 명령들이 32bit 의 길이 한계를 차지해야 하기때문이다. 조건분기명령인 경우 다음명령의 유효주소는 명령내에 있는 16bit 의 직접값으로부터 얻어 진다. 16bit 의 값은 맨 아래자리비트다음에 두개의 0 을 첨가하고 부호를 확장하여 32bit 의 값으로 확장된다.

상대주소화방식인 경우에는 24bit 의 직접값（무조건분기명령） 혹은 14bit 의 직접값（조건분기명령)이 절대주소화방식에서와 같이 확장된다. 결과값은 현재명령과의 상대적인 위치관계를 정의하는 프로그람계수기에 더해 진다.

다른 하나의 조건분기주소화방식은 **간접주소화방식**이다. 이 방식은 다음명령의 유효주소를 련결등록기나 계수등록기로부터 얻는다. 이 경우에 계수등록기는 분기명령을 위한 주소를 보존하는데 리용된다. 이 등록기는 이미 앞에서 설명한바와 같이 순환고리내에서의 계수에 리용되기도 한다.

산수연산명령

옹근수연산인 경우 모든 연산수들은 등록기내에 혹은 명령의 한 부분요소로 포함되여야 한다. 등록기주소화방식을 리용하는 경우 원천 혹은 목적연산수로는 일반목적등록기들중 어느 하나를 리용한다. 직접값주소화방식을 리용하는 경우 원천연산수는 명령에 16bit 의 부호 있는 량으로서 나타난다.

류점수연산에서는 모든 연산수들이 류점수등록기안에 있어야 하며 등록기주소화방식만을 리용할수 있다.

제 3 절. 명령형식

명령형식은 그의 구성요소의 견지에서 보면 명령비트들의 지면배치를 규정한다고 볼수 있다. 명령형식은 반드시 하나의 조작코드와 암시적이든 명시적이든 령 혹은 그이상의 연산수를 포함해야 한다. 모든 명시적인 연산수는 제 1 절에서 고찰한 주소화방식들중 어느 하나를 리용하여 참조된다. 형식에서는 매 연산수에 대한 주소화방식을 암시적으로 혹은 명시적으로 반드시 지적해야 한다. 대다수 명령모임들에서는 하나이상의 명령형식을 리용한다.

명령형식의 설계는 복잡한 기술이며 현재까지 놀라운 발전을 이룩하였다. 이 절에서는 먼저 일부 설계를 리용하여 기본설계문제점들을 고찰하고 다음절에서 Pentium Ⅱ와 PowerPC 의 명령형식을 고찰한다.

1. 명령길이

명령모임설계에서 대부분 처음에 제기되는 기초적인 문제는 명령형식의 길이에 대한것이다. 명령길이를 어떻게 결정하는가에 따라 기억기크기, 기억기조작, 모선구조, CPU 복잡성, CPU 속도 등에 영향을 주거나 또 영향을 받게 된다. 아쎔블리언어프로그람작성자는 알수 있겠지만 명령길이를 어떻게 결정하는가 하는것은 콤퓨터의 능력이나 유연성에 큰 영향을 준다.

명령길이를 결정하는데서 중요한것은 강력한 명령저장고의 구축과 기억공간을 절약할데 대한 요구를 합리적으로 해결하는것이다. 프로그람작성자는 더 많은 조작코드와 연산수, 더 많은 주소화방식, 더 큰 주소범위를 요구한다. 더 많은 조작코드와 연산수가 보장되면 보다 쉽게 프로그람을 작성할수 있게 된다. 또한 보다 많은 주소화방식은 프로그람작성자가 표조작이나 여러 궤도분기와 같은 일부 기능들을 실현하는데서 보다 큰 유연성을 가진다. 프로그람작성자는 이외에도 더 큰 주기억기를 요구하며 보다 많은 가상기억기를 리용하고 더 큰 기억기범위에 대한 주소화를 요구한다. 이와 같은 모든것들(조작코드, 연산수, 주소화방식, 주소범위)은 비트들을 요구하며 따라서 명령길이가 보다 더 길어 지게 된다. 그러나 길이가 긴 명령은 실행에서 효과적이지 못하다. 실례로 32bit 명령은 16bit 명령보다 2배의 기억공간을 차지한다. 그런데 효과성은 2배보다 훨씬 더 작다.

이러한 호상관계를 초월하여 다음과 같은것을 고찰해 보자. 명령길이는 기억기전송길이(모선체계에서는 자료모선길이)와 같거나 그의 배수가 되게 한다. 한편 명령의 꺼내기주기동안에는 완전한(옹근수에 해당하는) 명령이 얻어 지지 않는다. 이것은 기억기전송속도와 관련된다. 즉 기억기전송속도가 처리장치속도보다 빠르지 못하기때문이다. 처리장치가 명령을 불러 내는것보다 더 빨리 명령을 실행할수 있게 되면 기억기의 속도가 문제로 된다. 이를 위한 한가지 해결방법은 캐쉬(제4장 제3절 참고)를 리용하는것이며 다른 한가지 방법은 보다 짧은 명령을 리용하는것이다. 16bit 명령은 32bit의 명령보다 2배나 빠른 속도로 명령을 불러 들일수 있지만 실행에서는 2배정도 빠르지 못하다.

이와 같은 상황에서도 명령길이는 보통 8bit로 리용되는 문자길이와 고점수길이의 배수가 되도록 하여야 한다. 이를 위하여 불충분하게 정의된 용어인 **단어**(WORD)라는 개념을 리용할수 있다[FRAI83]. 기억기의 단어길이는 콤퓨터구성의 《본성적이며 고유한》단위로 본다. 보통 단어의 크기는 고점수의 크기(일반적으로 이 둘은 같다.)를 결정한다. 단어크기는 또한 기억기전송크기와 같거나 혹은 적어도 그의 옹근수배가 되여야 한다. 가장 일반적인 자료의 양식은 문자자료이므로 한 단어는 문자의 옹근수배를 기억한다. 한편 여러문자를 기억하거나 한 문자가 단어경계를 벗어 나는 경우 매 단어에는 쓸모 없는 비트들이 있게 된다.

2. 비트들의 할당

앞에서는 명령형식의 길이를 결정하는데 관계되는 일부 인자들에 대하여 보았다. 여기서 고찰하는것은 명령형식에서 비트들의 할당을 어떻게 하는가 하는것이다. 이것을 합리적으로 할당하는 문제도 역시 복잡한 문제이다.

명령길이가 주어 진 조건에서는 조작코드의 수와 주소화능력사이관계를 합리적으로 고려하여 비트들을 할당하여야 한다. 조작코드를 길게 하면(연산의 종류가 많아 진다.) 명백히 조작코드마당에 더 많은 비트가 할당되게 된다. 명령형식의 길이가 주어 진 조건에서 이것은 주소화에 리용할수 있는 비트수를 감소시키게 된다. 이 문제를 해결할수 있는 한가지 흥미 있는 방도는 가변길이조작코드를 리용하는것이다. 이것을 리용하면 조작코드길이를 최소로 할수 있다. 그러나 일부 조작코드에 대하여서는 명령안에 추가적인 비트를 리용하여 추가적인 연산을 규정해야 한다. 고정길이명령에서는 이렇게 하면 보다 적은 비트들이 주소화에 리용된다. 따라서 이 특징은 보다 적은 연산수와 보다 강력하지 못한 주소화를 요구하는 명령들에 리용된다.

주소화비트들의 리용을 결정하는데는 다음과 같은 인자들이 호상 련관되여 작용한다.

- **주소화방식의 수**: 때때로 주소화방식은 암시적으로 지정될수 있다. 실례로 어떤 조작코드들은 늘 첨수주소화에만 리용한다. 뿐만아니라 주소화방식은 명시적이여야 하며 이때 하나 혹은 그이상의 방식비트들이 요구된다.
- **연산수의 개수**: 주소의 개수가 작으면 작을수록 보다 길고 보다 다루기 힘든 프로그람이 나온다는것은 이미 고찰하였다(그림 9-3). 오늘날 콤퓨터에서 일반적인 명령들은 두개의 연산수를 리용하고 있다. 명령에서 매 연산수주소는 그 자신의 방식지적자를 요구하거나 혹은 이 방식지적자의 리용이 주소마당의 리용으로 제한될수 있다.
- **기억용등록기**: 콤퓨터는 처리를 위하여 CPU 안으로 들어 오는 자료를 보존하기 위한 등록기들을 반드시 가지고 있어야 한다. 한명의 사용자만이 볼수 있는 등록기 (보통 축적기라고 한다.)를 리용하는 경우 한 연산수주소는 암시적으로 되며 따라서 여기에 명령비트를 소비할 필요가 없다. 그러나 단일한 등록기에 의한 프로그람작성은 불편하며 많은 명령을 요구한다. 여러개의 등록기를 리용하는 경우는 몇개의 비트들이 등록기지정에 요구된다. 연산수참조에 등록기를 많이 리용하면 할수록 더 적은 비트들이 등록기지정에 리용된다. 많은 연구결과들은 사용자가 볼수 있는 등록기의 수가 대체로 8～32 개라는것을 보여 주고 있다 [LUND77, HUCK83].
- **등록기묶음의 수**: 많은 콤퓨터들은 하나의 일반목적등록기모임을 가지는데 이 모임은 8 혹은 16 개의 등록기들로 구성되여 있다.이 등록기들은 자료를 기억하거나 변위주소화를 위한 주소를 기억하는데 리용된다. 최근에는 일반목적등록기의 한 그룹이 아니라 둘 혹은 그이상의 특별한 묶음들의 집합체 (자료나 변위와 같은) 방향으로 이것을 실현하고 있다. 이 경향은 한소편극소형처리장치로부터 초대형콤퓨터에 이르기까지 모든 콤퓨터들에서 찾아 볼수 있다. 이 방법의 한가지 우점은 명령형식에서 얼마 안되는 비트들을 고정된 수의 등록기기능분할에 리용한다는것이다. 실례로 8 개의 등록기로 된 두개의 묶음에서는 매 등록기를 식별하는데 3 개의 비트만이 요구된다. 그것은 조작코드가 등록기묶음이 참조되고 있다는것을 암시적으로 지정하고 있기때문이다. 이 방법에도 한가지 부족점이 있다[LUND77]. 하나의 일반목적등록기모임을 가지고 있는 S/370 과 같은 체계에서 프로그람작성자는 보통 자료 및 변위를 위하여 등록기들을 각각 절반씩 할당하는 규칙을 적용하여 고정된 배치를 유지한다 [MALL79].
- **주소범위**: 기억기참조주소에서 참조할수 있는 주소의 범위는 주소비트의 수에 관계된다. 이것은 제한성으로 하여 직접주소화에 드물게 리용된다. 변위주소화에서 범위는 주소등록기의 길이에 의하여 결정된다. 등록기주소로부터의 큰 변위를 줄수만 있다면 주소화가 더 편리하겠지만 이렇게 되면 명령안에 상대적으로 큰 수의 주소비트들이 요구된다.
- **주소립도**: 등록기가 아니라 기억기를 참조하는 주소를 규정하는 다른 하나의 인자는 주소화의 립도이다. 16 혹은 32bit 의 단어를 취급하는 체계에서 주소는 설계가의 의도에 따라 한 단어 혹은 한바이트를 참조할수 있다. 바이트주소화는 문자조작에 편리하지만 고정된 크기의 기억기인 경우 더 많은 주소비트를 요구한다.

이상에서 고찰한바와 같이 설계가가 깊이 생각하여 균형을 맞추어야 할 많은 인자들이 있다. 여기서 어느 인자를 선택하는가 하는것은 명백하지 않다. 실례로 여러가지 명령형식방법들을 비교한 연구[CRAG79]에는 탄창의 리용, 일반등록기, 축적기, 기억기대 등록기방법들이 포함되여 있다. 이 실례에서도 일치한 가설묶음을 리용하면 코드공간

이나 실행시간에서의 큰 차이는 없다는것을 알수 있다.

이제 대표적인 두가지 콤퓨터들에서 이와 같은 여러가지 인자들의 균형을 어떻게 맞추는가를 간단히 보기로 하자.

PDP-8

일반용콤퓨터를 위한 한가지 가장 단순한 명령설계는 PDP-8 의 명령설계이다 [BELL78b]. PDP-8 은 12bit 명령을 리용하며 12bit 의 단어로 연산을 한다. 또한 단일일반등록기인 축적기도 가지고 있다.

이 설계에는 제한성이 있지만 주소화가 매우 유연하다. 모든 기억기참조는 7 개의 비트와 2개의 1bit변경자로 이루어 진다. 기억기는 매 페지가 $2^7 = 128$ 단어로 된 고정길이페지로 분할된다. 주소계산은 0 페지에 대한 참조 혹은 페지비트에 의해 결정되는 현재페지(이 명령을 포함하는 페지)에 대한 참조에 기초하여 진행한다. 두번째 수정자비트는 직접 혹은 간접주소화중 어느것이 리용되고 있는가를 지시한다. 이 두 방식은 간접주소가 페지 0 혹은 한 페지의 한 단어로 되는 12bit 의 주소가 되도록 결합하여 리용할수 있다. 페지 0에 있는 8개의 전용단어들은 자동첨수《등록기들》이다. 간접참조가 이 위치들중의 어느 하나에 의하여 진행되면 앞첨수지정이 일어 난다.

그림 10-4에서는 PDP-8의 명령형식을 보여 준다. PDP-8에는 3bit의 조작코드와 세가지 형태의 명령이 있다. 조작코드 0 부터 5 까지는 그 형식이 페지비트와 간접비트를 포함한 단일주소기억기참조명령이라는것을 보여 준다. 따라서 6 개의 기본연산밖에 존재하지 않는다는것을 알수 있다. 연산범위를 확장하기 위하여 조작코드 7 은 등록기참조 혹은 **마이크로명령**을 정의한다. 이 형식에서 나머지비트들은 추가적인 연산을 코드화하는데 리용된다. 이 비트들은 단일명령으로 결합될수 있다. 마이크로명령전략은 이미 DEC 에 의하여 PDP-1 에서 리용되고 있으며 PDP-1 은 오늘날 마이크로프로그람콤퓨터들의 선구자로 되고 있다. 조작코드 6은 I/O연산을 가리킨다. 이중에서 6개의 비트들은 64개의 장치들중 어느 하나를 선택하는데 리용되며 3개의 비트들은 알맞는 I/O지령을 규정한다.

PDP-8 명령형식은 매우 효과적이다. 이것은 간접주소화, 변위주소화, 첨수주소화를 모두 지원한다. 조작코드를 확장하면 근사적으로 총 35 개의 명령을 쓸수 있다. 12bit 길이로 명령이 제한되므로 이이상 더 좋게 만들수는 없다.

PDP-10

PDP-8명령모임과의 정확한 대비는 PDP-10명령모임을 통해서 할수 있다. PDP-10은 체계가 프로그람을 쉽게 만들도록 그 기능이 강화된 대규모시간공유체계로 설계되였다.

명령모임설계에 리용된 설계원리들은 다음과 같다[BELL78C].

- **직교성**: 직교성은 두 변수가 서로 독립이라는 원리이다. 명령모임의 문맥에서 조항은 명령의 다른 요소들이 조작코드와 독립이라는것을 의미한다. 즉 조작코드에 의해서 결정되지 않는다는것을 가리킨다. PDP-10설계가들은 조작코드와는 독립적인 꼭 같은 방법으로 주소가 계산된다는것을 나타내는 조항을 리용하여 명령모임을 설계하였다. 이것은 많은 콤퓨터들과는 대조되는것으로서 주소방식은 때때로 리용되는 연산자에 암시적으로 관계된다.
- **완전성**: 모든 연산자료형(옹근수, 고점수, 실수)은 완전하면서도 구별할수 있는 연산모임을 가져야 한다.
- **직접주소화**: 프로그람작성자에게 기억기조작의 부담을 주는 기준+변위주소화는 리용하지 않고 직접주소화방식을 리용한다.

이 모든 원리들은 안전한 프로그람작성의 기본목적을 달성하게 한다.

그림 10-4. PDP-8 명령형식

PDP-10 은 36bit 단어길이와 36bit 명령길이를 가진다. PDP-10 의 고정된 명령형식을 그림 10-5에 보여 주었다. 조작코드는 9개의 비트를 차지하며 512개의 연산을 규정할수 있다. 실지로는 총 365개의 명령이 정의되여 있다. 대다수 명령들은 2개의 주소를 가지며 이중에서 하나는 16 개의 일반등록기들중 어느 하나이다. 따라서 이 연산수참조는 4개의 비트를 차지한다. 다른 연산수참조는 18bit 의 기억기주소마당에서 출발한다. 이 마당은 직접값연산수 혹은 기억기주소로서 리용될수 있다. 기억기주소로 리용되는 경우 첨수주소화 및 간접주소화방식을 리용할수 있다. 첨수등록기로서는 일반등록기들을 리용한다.

36bit 의 명령길이는 매우 합리적이다. 그림 10-5에서 보여 준 조작코드보다 더 많은 조작코드를 가질 필요는 없다. 9 bit 의 조작코드마당이면 완전히 충분하다. 주소화 역시 단순하며 충분하다. 118bit 의 주소마당이면 필요한 직접주소화를 모두 실현할수 있다. 2^{18} 보다 기억기크기가 더 큰 경우에는 간접주소화를 리용한다. 또한 프로그람작성자의 편의를 도모하기 위하여 표작성이나 반복프로그람에 첨수주소화를 리용할수 있다. 18bit 연산수마당을 가지게 되면 직접값주소화가 합리적인것으로 된다.

그림 10-5. PDP-10 명령형식

PDP-10 명령모임설계는 초기에 목록화된 대상들에 알맞는다[LUND77]. PDP-10 명령모임은 기억공간을 비효과적으로 리용하여 소비하므로 프로그람작성자 혹은 콤파일러가 과제를 쉽게 수행할수 있게 한다. 이것은 설계가가 만들어 낸 의식적인것으로서 한심한 설계라고는 볼수 없다.

3. 가변길이명령

지금까지 고찰한 실례들에서는 단일고정명령길이를 리용하였으며 명령형식의 전후관계에서 합리적인 방안을 암시적으로 론의하였다. 설계가는 여러가지 길이를 가진 각이한 명령형식을 제공하는 대신에 그것들을 선택할수도 있다. 이것은 각이한 길이를 가진 조작코드의 큰 저장고를 쉽게 얻을수 있다. 주소화는 등록기와 기억기참조의 각이한 결합에 또 주소화방식들을 첨부하면 보다 유연하게 할수 있다. 가변길이명령을 리용하면 이와 같은 많은 변량들이 효과적으로 함축되여 주어 질수 있다.

가변길이명령은 CPU 의 복잡성을 증대시킨다. 하드웨어가격이 떨어 짐으로써 마이크로프로그람(제 4 편에서 론의됨)의 리용이나 CPU 설계원리를 리해하는데 모두 적은 가격이 들게 되였다.

가변길이명령의 리용은 모든 명령의 길이가 단어길이의 옹근수배가 되여야 한다는 요구를 무시하지 않는다. CPU 는 불러 내야 할 다음명령의 길이를 모르므로 대체로 가장 긴 명령의 길이와 같은 몇개의 바이트 혹은 단어를 불러 들인다. 이것은 때로는 여러개의 명령을 불러 들인다는것을 의미한다. 앞으로 제 11 장에서 고찰하겠지만 이것은 모든 경우에 알맞는 좋은 전략이다.

PDP-11

PDP-11 은 16bit 소형콤퓨터들의 제한범위에서 강력하고도 유연한 명령모임을 제공하도록 설계되였다[BELL70].

PDP-11 ㄱ는 8개의 16bit일반등록기들을 한 묶음으로 가지고 있다. 이 등록기들중에서 두개의 등록기는 부가적인 의미를 가진다. 하나는 특별한 목적의 탄창연산에 리용되는 탄창지시기이며 다른 하나는 다음명령의 주소를 가지는 프로그람계수기이다.

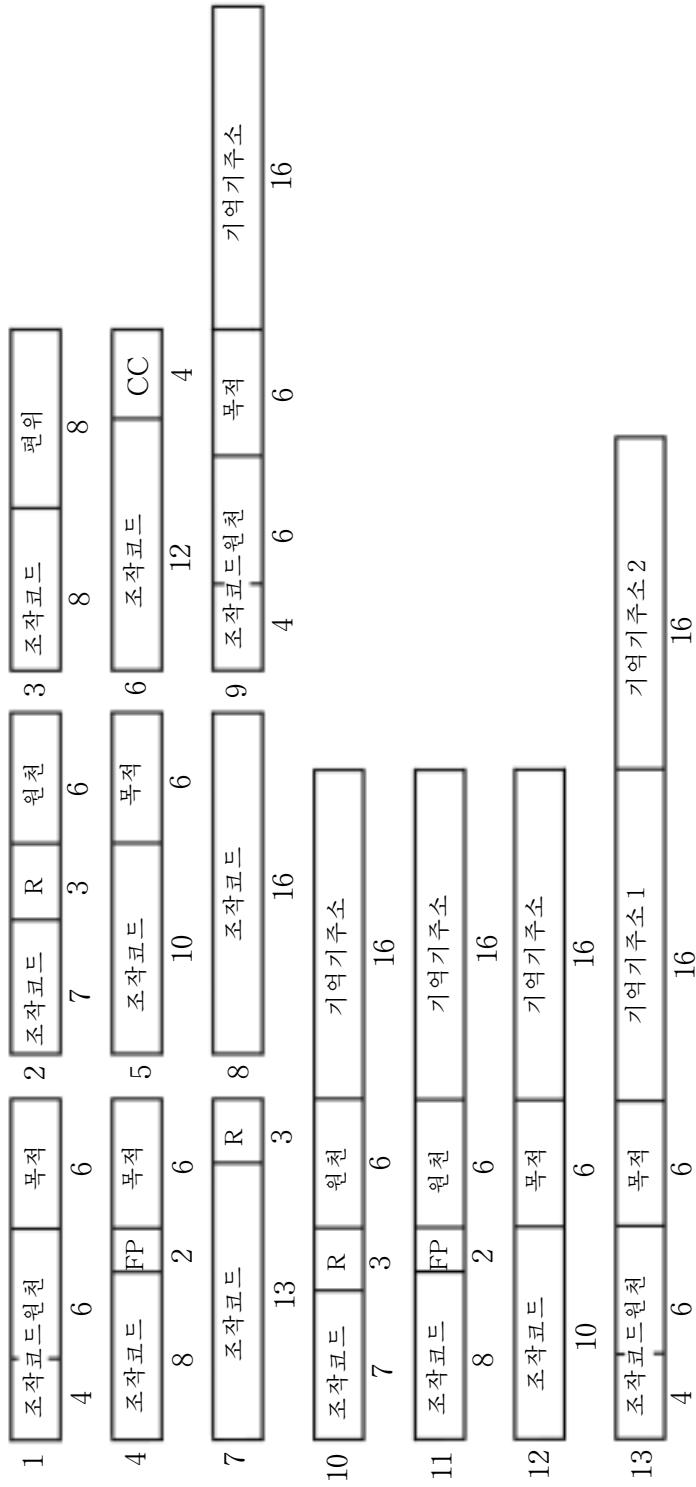
그림 10-6 은 PDP-11 의 명령형식을 보여 주고 있다. 13 개의 서로 다른 명령형식들이 있는데 이것들은 0, 1, 2 주소화명령형식을 취하고 있다. 조작코드는 그 길이가 4 로부터 16bit 까지 변할수 있다. 등록기참조는 길이가 6bit 이다. 이중에서 3 개의 비트는 등록기를 식별하는데 리용하며 나머지 3bit 는 주소화방식을 규정한다. PDP-11 은 풍부한 주소화방식을 가지고 있다. 조작코드가 아니라 연산수로 주소화방식을 련결하는것의 한가지 우점은 임의의 주소화방식을 조작코드로 리용할수 있는것이다. 이미 언급한바와 같이 이 독립성을 **직교성**이라고 한다.

원천 및 목적은 모두 3-bit 의 주소화방식마당과 3-bit 의 등록기수마당을 가진다.
FP 는 류동소수점수등록기 0,1,2,3 에 대하여 1 이다.
R 는 일반등록기에 대하여 1 이다.
CC 는 조건코드마당이다.

그림 10-6. PDP-11 의 명령형식(수자들은 마당길이를 표시)

PDP-11 명령은 보통 긴 하나의 단어(16bit)이다. 일부 명령에서는 하나 혹은 2개의 기억기주소가 첨부되여 32 혹은 48bit 의 명령이 저장고의 한 부분으로 된다. 이것은 주소화에서 더 큰 유연성을 보장한다.

PDP-11 명령모임과 주소화능력은 복잡하다. 이것은 하드웨어의 비용과 프로그람작성의 복잡성을 증가시킨다. 우점은 보다 효과적이며 함축된 프로그람을 개발할수 있다는것이다.

VAX

대다수 콤퓨터구성방식은 상대적으로 작은 수의 고정된 명령형식을 준다. 이 명령형식은 프로그람작성자들에게 다음과 같은 두가지 문제를 야기시킬수 있다. 첫째로, 주소화방식과 조작코드가 직교하지 않는다는것이다. 실례로 어떤 주어 진 연산에서 한 연산수는 등록기, 다른 연산수는 기억기이거나 혹은 둘 다 등록기 기타 등으로 되여야 한다. 둘째로, 제한된 수의 연산수들만이 리용될수 있다는것이다. 즉 둘 혹은 셋까지의 연산수를 리용할수 있다. 일부 연산들은 보다 많은 연산수를 요구하므로 여러가지 방법들을 둘 혹은 그 이상의 명령을 리용하여 필요한 결과를 얻는데 리용해야 한다.

이 문제를 풀기 위하여 다음과 같은 두가지 준칙이 VAX 명령형식을 설계하는데 리용되였다[STRE78].

1. 모든 명령은 《고유하면서도 본성적인》수의 연산수를 가져야 한다.
2. 모든 연산수는 그 규정에서 똑 같은 일반성을 가져야 한다.

이것은 매우 가변적인 명령형식을 낳는다. 명령은 조작코드에 따라 0～6 개의 연산수지적자를 가진 1 혹은 2byte 의 조작코드로 이루어 진다. 최소명령길이는 1byte 이며 37byte 까지의 명령을 만들수 있다. 그림 10-7 에 몇가지 실례를 보여 주었다.

VAX 명령은 한바이트의 조작코드로 시작한다. 한바이트의 조작코드이면 대부분 VAX 명령을 조종하기에는 충분하다. 그러나 300 개이상의 각이한 명령이 있으므로 8bit 를 가지고서는 불충분하다. 16 진코드 FD 와 FF 는 두번째 바이트에서 규정되는 실지 조작코드와 함께 확장된 조작코드를 지시한다.

명령의 나머지부분은 최대로 6 개의 연산수지적자들로 구성된다. 연산수지적자는 왼쪽에서부터 4 개의 비트들이 주소방식지적자인 최소 1byte 형식을 가진다. 이 규칙에 맞지 않는 경우는 문자방식으로서 이 방식은 6bit 의 문자를 위한 공간을 남기면서 제일 왼쪽에 있는 2개의 비트로 패턴 00을 신호한다. 이와 같은 제외로 인하여 총 12개의 각이한 주소화방식을 규정할수 있다.

연산수지적자는 보통 한바이트로 구성되며 여기서 오른쪽 4 개의 비트들은 16 개의 일반목적등록기들중 어느 하나를 규정한다. 연산수지적자의 길이는 두가지 방법들중 어느 하나로 확장할수 있다. 첫번째 방법은 하나 혹은 그이상 바이트들의 상수값이 연산수지적자의 첫번째 바이트를 즉시 뒤따르게 하는것이다. 이 방법의 실례는 변위방식이며 이 방식에서는 8, 16 혹은 32bit 의 변위를 리용한다. 두번째 방법은 첨수방식의 주소화를 리용하는것이다. 이 경우에 연산수지적자의 첫번째 바이트는 4bit 의 주소화방식코드 0100 과 4bit 의 등록기식별자로 구성된다. 연산수지적자의 나머지부분은 길이가 하나 혹은 그이상의 바이트가 될수 있는 기준주소지적자로 된다.

독자들은 어떤 종류의 명령이 6 개의 연산수를 요구하는가 하고 이상하게 생각하면

서 놀랄수 있다. 놀랍게도 VAX는 얼마간의 이러한 명령들을 가지고 있다. 이제 명령

ADDP6 OP1, OP2, OP3, OP4, OP5, OP6

을 고찰해 보자. 이 명령은 2개의 조임형 10진수를 더하는 명령이다. OP1과 OP2은 하나의 10 진수렬의 길이와 시작주소를 규정하며 OP3과 OP4는 두번째 10 진수렬의 길이 및 시작주소이다. 두개의 10 진렬은 서로 더해져 결과가 OP5와 OP6에 의해 규정되는 길이와 시작위치를 가지는 10진수렬에 기억된다.

그림 10-7. VAX명령의 실례

VAX 명령모임은 다양한 연산과 주소화방식을 가지고 있다. 이것은 콤파일작성도구와 같은 프로그람을 개발하기 위한 매우 강력하면서도 유연한 도구를 프로그람작성자들에게 제공해 준다. 리론적으로 이것은 고급언어프로그람의 효과적인 기계어번역을 실현하며 일반적으로 효과적이며 능률적으로 CPU 자원을 리용할수 있게 한다. 이와 같은 리득이 있는 반면에 보다 단순한 명령모임과 형식을 가진 처리장치에 비하면 CPU의 복잡

성이 증가하는 결함도 있다.

이에 대한 구체적인 고찰은 제12장에서 한다. 여기서는 매우 단순한 명령모임인 경우만을 대상으로 고찰한다.

제4절. Pentium Ⅱ와 PowerPC의 명령형식

1. Pentium Ⅱ의 명령형식

Pentium Ⅱ는 각이한 명령형식을 가지고 있다. 이 절에서 고찰하는 요소들중에서 조작코드마당만은 늘 나타난다. 그림 10-8에서는 명령형식을 보여 주었다. 명령은 0～4개까지의 선택적인 명령앞붙이, 하나 혹은 두바이트의 조작코드, 선택적인 주소지적자로 이루어 진다. 여기서 주소지적자는 Mod r/m 바이트, 척도첨수바이트, 선택적인 변위, 선택적인 직접값마당으로 이루어 진다.

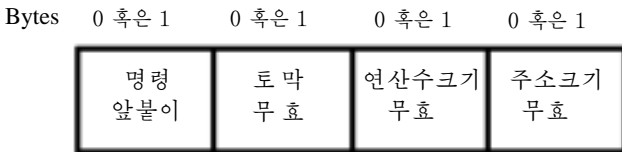
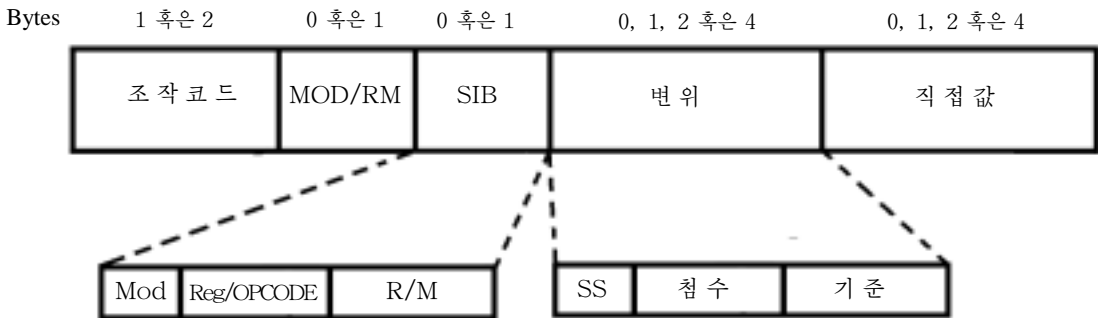



그림 10-8. Pentium Ⅱ의 명령형식
ㄱ-앞붙이, ㄴ-명령

먼저 앞붙이바이트들을 고찰해 보자.

- **명령앞붙이** : 명령앞붙이는 LOCK 앞붙이 혹은 반복앞붙이들중 어느 하나로 구성된다. LOCK 앞붙이는 다중처리장치환경에서 공유기억기의 전문리용을 확보하는데 리용된다. 반복앞붙이는 Pentium Ⅱ가 규칙적인 쏘프트웨어순환고리에서보다 훨씬 더 빨리 렬을 처리할수 있게 하는 렬의 반복연산을 규정한다. 여기에는 5개의 서로 다른 앞붙이가 있다. 즉 REP, REPE, REPZ, REPNE와 REPZ 등이다. 절대 REP 앞붙이가 존재하면 명령에서 규정된 연산은 련속되는 렬요소에 대하여

반복실행된다. 반복실행수는 등록기 CX 로 규정한다. 조건 REP 앞붙이는 명령이 CX 의 내용이 0 으로 될 때까지 혹은 조건이 만족될 때까지 계속 반복되게 한다.

- **토막표시** : 명령이 어느 토막등록기를 리용하는가를 명시적으로 규정한다. 이때 이 명령을 위하여 발생되는 암시적인 토막등록기선택은 무시된다.
- **주소크기** : 처리장치는 16 혹은 32bit 의 주소를 리용하여 기억기를 지정할수 있다. 주소의 크기는 명령에서 변위크기를 결정하며 유효주소계산기간에 발생한 주소의 변위크기를 규정한다. 이 크기들중 하나는 암시적으로 지적되며 주소크기앞붙이는 32bit 와 16bit 주소발생사이에서 절환한다.
- **연산수크기** : 명령은 16 혹은 32bit 의 암시적인 연산수크기를 가진다. 또한 연산수앞붙이는 32bit 와 16bit 연산수사이에서 절환한다.

명령 그자체는 다음과 같은 마당을 포함하고 있다.

- **조작코드** : 하나 혹은 두바이트의 조작코드. 조작코드는 자료가 바이트크기 혹은 완전크기(문맥에 따라 16 혹은 32bit)인 경우와 자료연산의 방향(기억기에로 혹은 기억기로부터), 직접값자료마당이 부호확장된것인지 아닌지를 규정하는 비트들을 가지고 있다.
- **Mod r/m**: 이 바이트와 그다음바이트는 주소화정보를 규정한다. Mod r/m 바이트는 연산수가 등록기라는것을 지정하며 기억기내에 있는 경우 바이트안의 마당들을 리용하는 주소화방식을 규정한다. Mod r/m 바이트는 세개의 마당으로 구성된다. Mod 마당(2 bit)은 32 개의 가능한 값들 즉 8 개의 등록기와 24 개의 첨수지정방식을 이루는 r/m 마당과 결합한다. Reg/Opcode 마당(3bit)은 등록기수를 규정하거나 3bit 이상의 마당으로 조작코드정보를 규정한다. r/m 마당(3 bit)은 연산수위치로써 등록기를 지정할수 있으며 또 Mod 마당과 결합하여 주소화방식의 한 부분으로서의 역할을 한다.
- SIB: Mod r/m 바이트의 일정한 코드화는 주소화방식을 완전히 규정하는 SIB 바이트를 규정한다. SIB 바이트는 3 개의 마당으로 구성된다. SS 마당(2bit)은 척도첨수주소화를 위한 척도인자를 규정하며 첨수마당(3bit)은 첨수등록기를, 기준마당(3bit)은 기준등록기를 규정한다.
- **변위** : 주소화방식지적자가 변위방식을 리용하고 있는 경우 8, 16 혹은 32bit 의 부호있는 옹근수 변위마당을 추가한다.
- **직접값**: 8, 16 혹은 32bit 의 연산수값을 규정한다.

아래에서는 여러가지 명령형식의 비교를 진행한다. Pentium II 형식에서 주소화방식은 매 연산수에서가 아니라 조작코드렬의 한 부분으로서 주어 지게 된다. 주소화방식정보는 하나의 연산수만이 가질수 있으므로 명령에서는 하나의 기억기만을 참조할수 있다. 그러나 VAX 방식에서는 모든 연산수가 주소화방식정보를 가지고 있다. 이것은 기억기대 기억기연산을 가능하게 한다. 결국 Pentium II 명령들은 VAX 명령에 비해 보다 압축되여 있다. 그러나 기억기 대 기억기연산을 해야 하는 경우 VAX 는 이것을 단일한 명령으로 실행할수 있다.

Pentium II 형식은 1byte 만이 아니라 2 byte 와 4 byte 의 첨수주소화용변위를 리용할수

있다. 보다 큰 첨수변위의 리용은 보다 긴 명령을 낳지만 그래도 이 특징은 유연성을 보장한다. 실례로 큰 배렬 혹은 큰 탄창을 주소화하는데 쓸모가 있다. 그러나 IBM S/370 명령형식은 4Kbyte(12bit 의 변위정보)보다 더 큰 정보는 허용하지 않으며 변위가 반드시 정수여야 한다. 위치가 이 변위내에 있지 않는 경우 콤파일러는 요구되는 주소로 범위초과코드를 발생시킨다. 이 문제는 4Kbyte 를 초과하여 차지하는 국부변수를 가지는 탄창틀로서 취급하면 아주 명백해 진다.

이상에서 고찰한바와 같이 Pentium II 명령의 코드화는 매우 복잡하다. 코드화는 8086 콤퓨터와 역방향호환성이 보장되여야 한다는 요구 그리고 효과적인 코드를 만들도록 콤파일러작성도구에서 모든 가능한 방조를 제공할데 대한 일부 설계가들의 요구를 만족시키도록 부분적으로 해야 한다. 이와 같이 복잡한 명령모임을 RISC 명령모임의 상반되는 극단으로 보는것이 오히려 더 좋은지는 론의해 보아야 할 문제이다.

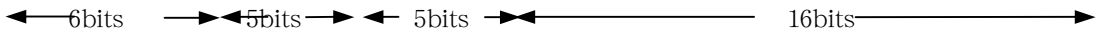
2. PowerPC 의 명령형식

Power PC 의 모든 명령은 32bit 길이를 가지며 규칙적인 형식을 가지고 있다. 명령에서 첫 6 개의 비트들은 수행되는 연산을 규정한다. 일부 경우에는 연산의 특별한 부분경우를 규정하도록 명령밖으로 조작코드를 확장한다. 그림 10-9에서 그늘진 부분은 조작코드비트를 표시한다.

명령형식의 규칙적인 구조는 명령실행장치의 일감을 덜어 준다. 모든 넣기/기억 및 산수연산과 론리연산명령들에서 조작코드는 32 개의 일반목적등록기를 리용할수 있도록 두개의 5bit 등록기참조를 동반하게 된다.

분기명령은 분기명령에 뒤따르는 명령의 유효주소가 련결등록기에 있다는것을 지시하는 련결(L)비트를 포함하고 있다. 두 양식의 명령은 주소화방식이 절대주소화방식인가 혹은 프로그람계수기상대방식인가를 지시하는 비트 (A) 를 가지고 있다. 조건분기명령에서 CR 비트마당은 조건등록기에서 검사되는 비트를 지정한다. 선택마당은 분기가 어떤 조건하에서 취해 지는가 하는 조건을 규정한다. 조건에는 다음과 같은것들이 있다.

- 늘 분기
- 계수값이 0 이 아니고 조건이 거짓이면 분기
- 계수값이 0 이 아니고 조건이 참이면 분기
- 계수값이 0 이고 조건이 거짓이면 분기
- 계수값이 0 이고 조건이 참이면 분기
- 계수값이 0 이 아니면 분기
- 계수값이 0 이면 분기
- 조건이 거짓이면 분기
- 조건이 참이면 분기

계산(산수연산, 류점수연산, 론리연산 등)을 진행하는 대다수 명령들은 연산결과를 조건등록기에 기록하겠는가를 지시하는 하나의 비트를 가지고 있다. 이 특징은 분기예측처리에 효과적으로 리용할수 있다.

6bits	5bits	5bits	16bits		
분기	긴 직접값			A	L
조건분기	Options	CR Bit	변위분기	A	L
조건분기	Options	CR Bit	련결 혹은 계수등록기를 통한 간접		L

ㄱ)

CR	목적비트	원천비트	원천비트	Add, Or, Xor, 기타	/

ㄴ)

넣기/기억(간접)	목적등록기	기준등록기	변위		
넣기/기억(간접)	목적등록기	기준등록기	첨수등록기	크기, 부호, 갱신	/
넣기/기억(간접)	목적등록기	기준등록기	변위		XO *

ㄷ)

산수연산	목적등록기	Src 등록기	Src 등록기	O	Add, Sub, 기타		R
Add, Sub, 기타	목적등록기	Src 등록기	부호 있는 직접값				
론리연산	Src 등록기	목적등록기	Src 등록기	Add, Or, Xor, 기타			R
And, Or, 기타	Src 등록기	목적등록기	부호 없는 직접값				
회전	Src 등록기	목적등록기	Amt 만큼 밀기	마스크 시작	마스크 끝		R
회전 혹은 밀기	Src 등록기	목적등록기	Src 등록기	밀기형 혹은 마스크			R
회전	Src 등록기	목적등록기	Amt 만큼 밀기	마스크	XO	S	R *
회전	Src 등록기	목적등록기	Src 등록기	마스크	XO		R *
밀기	Src 등록기	목적등록기	Amt 만큼 밀기	밀기형 혹은 마스크		S	R *

ㄹ)

류점수 단정확도/배정확도	목적등록기	Src 등록기	Src 등록기	Src 등록기	류점수더하기, 기타	R

ㅁ)

A - 절대 혹은 PC 상대
L - 보조루틴에 련결
O - XER 에서 레코드자리넘침
*- 64bit 에서만 실현

R - CR1 에서 레코드조건
XO - 조작코드확장
S - 모든 밀기량마당의 부분

그림 10-9. PowerPC 의 명령형식; ㄱ-분기명령, ㄴ-조건등록기론리명령, ㄷ-넣기/기억명령, ㄹ-옹근수산수연산, 론리연산, 밀기/회전명령, ㅁ-류점수산수연산명령

류점수명령은 세개의 원천등록기를 위한 마당을 가지고 있다. 많은 경우 2 개의 원천등록기만을 리용한다. 몇가지 명령들은 두 원천등록기의 곱하기를 진행하고 세번째 원천등록기의 더하기 혹은 덜기를 진행한다. 이 복합명령들은 같은 형의 연산이 계속 반복되여 진행되는 경우에 리용된다. 실례로 행렬연산의 한 부분인 내부적의 계산은 곱하기ー더하기를 리용하여 실현할수 있다.

참 고 문 헌

제 9 장에서 인용한 참고문헌들은 이 장의 참고문헌으로 리용할수도 있다. [BALL97]은 명령형식들의 상세한 론의와 주소화방식들을 고찰하고 있다. 또한 형식과 관련한 명령모임설계문제점들의 고찰을 위해서는 특별히 [FLYN85]을 참고하면 좋다.

BLAA97 Blaauw, G., and Brooks, F. *Computer Architecture: Concepts and Evolution.* Reading, MA: Addison-Wesley, 1997

FLYN85 Flynn, M.: Johnson, J.: and Wakefield, S. "On Instruction Sets and Their Formats." *IEEE Transactions on computers,* March 1985

련 습 문 제

1. 32bit 명령이 16bit 명령에 비해 쓸모가 2 배도 되지 않는다는것을 론증하시오.
2. 다음과 같은 기억기값과 축적기를 가진 1 주소콤퓨터가 주어 진 경우 다음명령들은 축적기에 어떤 값을 넣는가?
   - 단어 20 은 40 을 가지고 있다.
   - 단어 30 은 50 을 가지고 있다.
   - 단어 40 은 60 을 가지고 있다.
   - 단어 50 은 70 을 가지고 있다.

     ㄱ. LOAD IMMEDIATE 20
     ㄴ. LOAD DIRECT 20
     ㄷ. LOAD INDIRECT 20
     ㄹ. LOAD IMMIDIATE 30
     ㅁ. LOAD DIRECT 30
     ㅂ. LOAD INDIRECT 30
3. 프로그람계수기에 기억된 주소가 기호 X1 로 표시된다고 하자. X1 에 기억된 명령은 주소부분(연산수참조) X2 를 가진다. 명령실행에 요구되는 연산수는 주소 X3 에 기억기단어로 기억된다. 첨수등록기는 값 X4 를 가지고 있다. 이와 같은 경우 명령주소화방식이 ㄱ-직접, ㄴ-간접, ㄷ-PC 상대, ㄹ-첨수인 경우 이 여러가지 량들사

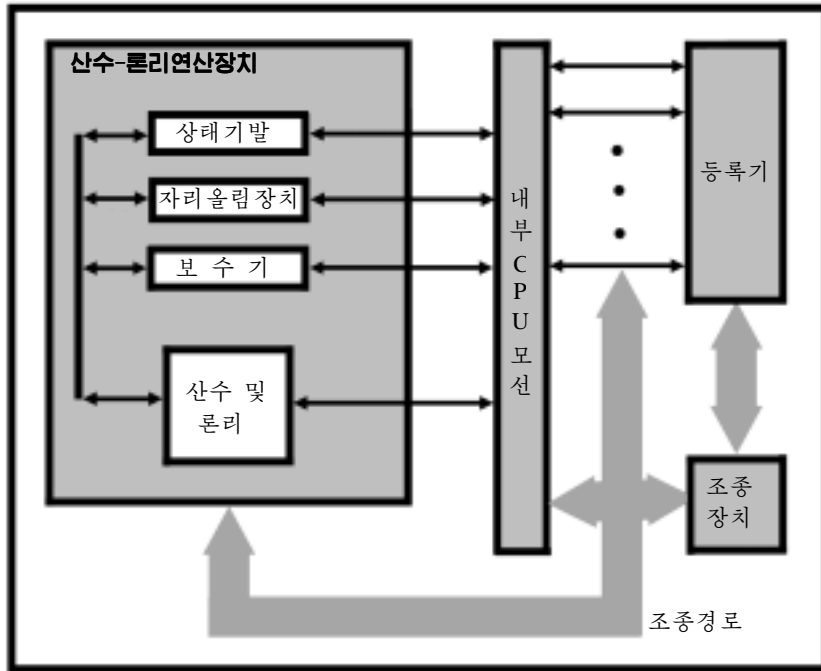
이 관계는 어떻게 되겠는가?

4. PC 상대방식의 분기명령은 주소 620_{10} 기억기에 기억되여 있다. 이 분기명령에 의하여 위치 530_{10} 으로 이행한다. 명령내의 주소마당은 10bit 로 길다. 명령내에서 그의 2 진수값은 얼마인가?
5. 간접주소화방식의 명령을 불러 내여 실행할 때 그 명령이 단일연산수를 요구하는 계산명령(ㄱ)이거나 분기명령(ㄴ)이라면 CPU 는 기억기참조를 몇번 요구하는가?
6. IBM370 은 간접주소화방식을 가지고 있지 않다. 연산수의 주소가 주기억기에 있다고 가정하면 연산수를 어떻게 호출하는가?
7. IBM 이 왜 단어단위를 36bit 로부터 32bit 로 바꾸었는가?
8. 참고문헌[COOK82]에서는 탄창리용을 비롯한 다른 주소화방식에 유리하도록 PC 상대주소화방식을 배제한다는것을 제안하고 있다. 이 제안의 부족점은 무엇인가?
9. 고정된 16bit 명령길이를 리용하는 명령모임이 있다고 하자. 연산수는 길이가 6 bit 이다. 또한 k 개의 두 연산수명령과 L 개의 령-연산수명령이 있다. 하나의 연산수를 가진 명령의 최대수는 얼마인가?
10. 다음의 모든 명령들이 36 bit 의 명령으로 부호화되도록 가변길이조작코드를 설계하시오.
    - 두개의 15 bit 주소와 한개의 3 bit 등록기번호를 가진 명령
    - 하나의 15 bit 주소와 하나의 3 bit 등록기번호를 가진 명령
    - 주소나 등록기를 가지지 않는 명령
11. 문제 9-3 의 결과를 고찰해 보자. M 이 16bit 기억기주소이고 X, Y 와 Z 가 16bit 주소 혹은 4bit 등록기주소라고 가정하자. 한바이트주소콤퓨터는 축적기를 리용하여 2, 3 주소콤퓨터를 16 개의 등록기와 기억기위치 및 등록기들의 모든 결합으로 동작하는 명령들을 가진다. 조작코드의 길이는 8bit 이고 명령길이가 4 의 배수라고 가정하면 매 명령은 X 를 계산하는데 몇 비트를 요구하는가?
12. 2 개의 조작코드를 가진 명령이 있다면 어떤 정당성이 가능한가?
13. Pentium Ⅱ 는 다음과 같은 명령을 가지고 있다.

    IMUL OP1 , OP2 , 직접값

    이 명령은 등록기나 기억기가 될수 있는 OP2 에 직접값을 곱하고 그 결과를 OP1 에 넣는데 OP1 은 등록기여야 한다. 명령모임에는 이 부류에 속하는 다른 3 연산수명령은 없다. 이러한 명령을 리용할수 있는 가능성은 무엇인가? (**암시** : 첨수지정을 생각해 보시오.)

제 11 장. CPU 구조와 기능

◆◆◆

◆ 처리장치는 사용자가 볼수 있는 등록기들과 조종 및 상태등록기들을 가지고 있다. 사용자가 볼수 있는 등록기들은 기계명령에서 암시적으로 혹은 명시적으로 참조될수 있다. 사용자가 볼수 있는 등록기들은 고정수 혹은 류점수, 주소, 첨수, 토막지시기 등과 같은 일반목적이나 전용으로 리용할수 있다. 조종 및 상태등록기는 CPU 의 연산을 조종하는데 리용된다. 대표적인 실례가 프로그람계수기이다. 다른 하나의 실례는 여러가지 상태 및 조건비트들을 가지는 프로그람상태단어（PSW）이다. 이것들은 마지막산수연산결과를 반영하는 비트들과 새치기가능비트들, CPU 가 체계관리자 혹은 사용자방식으로 동작하고 있다는것을 보여 주는 지시기를 포함한다.

◆ 처리장치는 실행속도를 높이기 위하여 명령관흐름처리를 리용한다. 관흐름처리는 명령주기를 여러개의 개별적인 단계로 갈라 놓은것을 말하는데 이 개별적인 단계는 다음과 같은 순서로 일어 난다. 즉 명령꺼내기, 명령해신, 연산수주소결정, 연산수꺼내기, 명령의 실행, 연산수결과의 쓰기와 같은 순서로 단계화된다. 명령은 조립관흐름에서와 같이 이 단계를 거치게 되며 원리적으로 매 단계는 같은 시간에 서로 다른 명령을 수행할수 있다. 분기의 발생과 명령들사이 의존성은 관흐름의 설계와 리용을 복잡하게 만든다.

◆◆◆

이 장에서는 제 3 편에서 아직까지 고찰하지 못한 처리장치의 개념을 보고 제 12 장과 제 13 장에서 RISC 와 슈퍼스칼라구성방식을 론의하기 위한 단계를 설정한다.

이 장에서는 처리장치구성에 대한 개괄로부터 시작한다. 그다음 처리장치의 내부등록기를 이루는 등록기들을 분석한다. 명령주기에 대한 고찰에서는 그 단계들과 관호름처리로서 알려 진 일반기술을 기본으로 고찰한다. 마지막으로 Pentium II 와 Power PC 조직에 대한 개념을 고찰한다.

제 1 절. 처리장치조직

CPU 의 조직을 리해하기 위하여 그것이 해야 할 일인 CPU 에 대한 요구를 고찰해 보자.

- **명령꺼내기** : CPU 는 기억기로부터 명령을 읽는다.
- **명령해석** : 명령은 그것이 어떤 작용을 요구하는 명령인가를 결정하도록 해신된다.
- **자료꺼내기** : 명령의 실행은 기억기 혹은 I/O 모듈로부터 자료를 읽을것을 요구한다.
- **자료처리** : 명령의 실행은 자료에 대한 일부 산수 및 론리연산을 수행할것을 요구한다.
- **자료쓰기** : 실행결과는 기억기 혹은 I/O 모듈에 자료를 쓸것을 요구한다.

이와 같은 일을 하자면 CPU 가 일부 자료를 림시로 기억해야 한다. CPU 는 다음명령을 어디에서 얻을수 있도록 마지막명령의 위치를 기억해야 한다. 또한 명령이 실행되고 있는 동안 명령과 자료를 림시로 기억해야 한다. 다시말하여 CPU 는 작은 내부기억기를 가져야 한다.

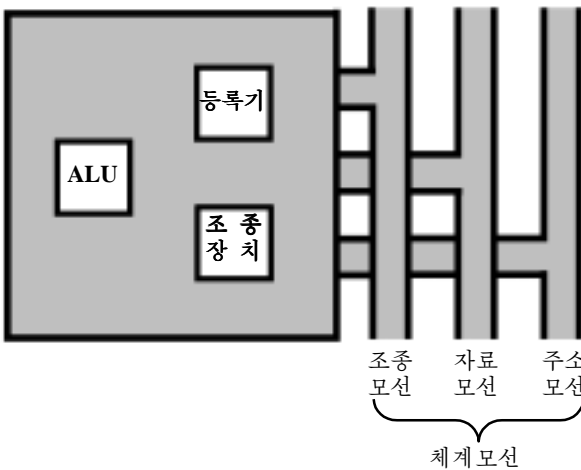



그림 11-1. 체계모선을 가진 CPU

그림 11-1 은 체계모선을 통하여 체계의 나머지부분과 련결을 보장하는 CPU 의 간략화된 구성을 보여주고 있다. 이미 제 3 장에서 서술한 이와 류사한 대면부는 임의의 호상련결구조에서 보편적인것으로 되고 있다. 독자들은 CPU 의 기본구성요소들이 **산수-론리연산장치**와 **조종장치**라는것을 상기할수 있다. ALU 는 실지 자료의 계산이나 처리를 한다. 조종장치는 CPU 의 안으로 혹은 밖으로 자료나 명령의 이동을 조종하며 ALU 의 연산을 조종한다. 그림 11-1 은 이외에도 이른바 **등록기**라고 하는 기억기위치들의 묶음으로 이루어 지는 최소내부기억기를 보여 주고 있다.

CPU 에 대한 약간 더 상세한 고찰은 그림 11-2 에서 보여 주고 있다. 자료전송 및 론리조종경로들이 표시되여 있으며 내부 CPU 모선이라고 하는 요소도 있다. 내부 CPU 모선은 각이한 등록기들과 ALU 사이의 자료전송을 위한것이다. 이것은 사실상 ALU 가 내부 CPU 기억기에 있는 자료에 대해서만 연산을 하기때문이다. 그림 11-2 에서는 또한

ALU의 대표적인 기본요소들도 보여 주고 있다. 이 그림으로부터 콤퓨터전체로서의 내부구조와 CPU 내부구조사이에 류사성이 있다는것을 알수 있다. 두 경우는 모두 자료경로에 의해 련결된 기본요소들(콤퓨터 : CPU , I/O, 기억기 ; CPU : 조종장치, ALU, 등록기)의 작은 집합체라는것을 알수 있다.

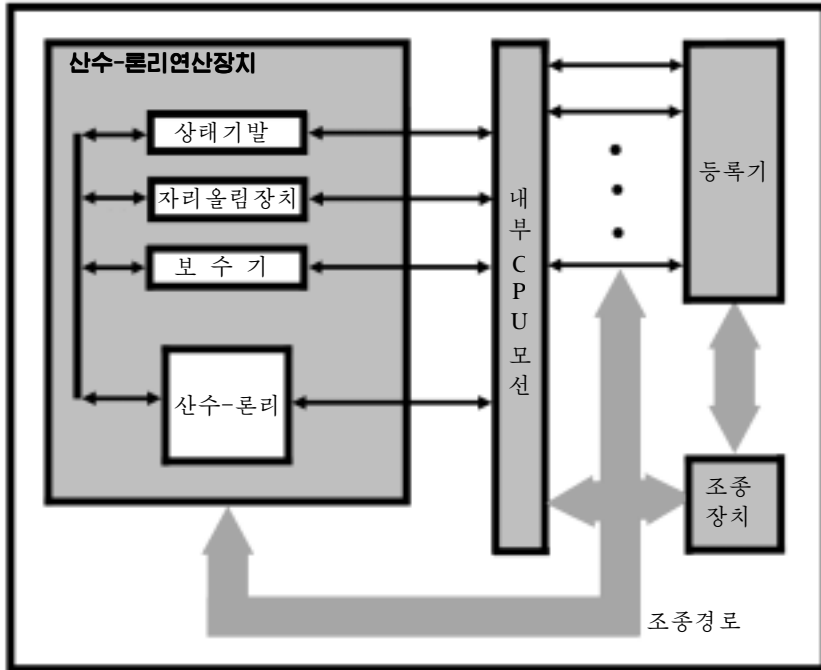



그림 11-2. CPU의 내부구조

제 2 절. 등록기조직

제 4 장에서 이미 론의한 바와 같이 콤퓨터체계는 기억기계층을 리용하고 있다. 기억기의 속도는 계층의 보다 높은 준위에서 더 빠르며 크기가 더 작고 비트당 더 많은 비용이 든다. CPU 안에는 기억기계층내에 주기억기와 캐쉬이상의 기억기준위의 기능을 수행하는 등록기묶음이 있다. CPU 안에 있는 이 등록기들은 다음과 같은 두가지 기능을 수행한다.

- **사용자가 볼수 있는 등록기** : 이 등록기들은 기계어 혹은 아쎔블리언어프로그람작성자가 등록기리용을 최적화하여 주기억기참조를 최소로 되게 한다
- **조종 및 상태등록기** : 이 등록기들은 CPU 의 연산을 조종하는 조종장치나 프로그람의 실행을 조종하는 특권조작체계프로그람에 의하여 리용된다.

이와 같은 기능을 수행하는 두 종류의 등록기들을 정확히 구별하기는 힘들다. 실례를 들면 일부 콤퓨터들에서 프로그람계수기는 사용자가 볼수 있는 등록기로 되지만(례로 VAX) 다른 많은 콤퓨터들에서는 그렇지 않다. 그러나 앞으로의 론의를 위하여 이와 같은 구분을 리용한다.

1. 사용자가 볼수 있는 등록기

사용자가 볼수 있는 등록기는 CPU 가 실행하는 기계어에서 참조할수 있는 등록기이다. 이 등록기들은 다음과 같이 분류할수 있다.

- 일반등록기
- 자료등록기
- 주소등록기
- 조건코드등록기

일반등록기들은 프로그람작성자에 의하여 여러가지 기능을 수행하도록 할당되는 등록기이다. 때때로 명령모임에서 이 등록기들은 연산과는 독립적으로 리용된다. 즉 임의의 일반등록기는 임의의 조작코드에 대한 연산수로 리용한다. 이것이 정확한 일반등록기의 리용이다. 그러나 보통은 제한이 있다. 실례로 류점수 및 탄창연산인 경우 전용등록기들을 리용한다.

일부 경우에 일반등록기들은 주소지정기능(례를 들어 등록기간접,변위주소지정)에 리용할수도 있다. 또 어떤 경우에는 자료등록기들과 주소등록기들사이의 부분적인 혹은 완전한 분리를 할수 있다. **자료등록기**들은 자료를 유지하는데만 리용될수 있으며 연산수주소계산에 리용할수는 없다. **주소등록기**들은 일부 일반등록기로 리용되기도 하며 특별한 주소지정방식에 전용으로 쓰이기도 한다. 주소등록기에는 다음과 같은것들이 있다.

- **토막지시기**: 토막주소지정(제 7 장 제 3 절 참고)을 가진 콤퓨터에서 토막등록기는 토막의 기준주소를 가진다. 여기에는 여러개의 등록기들이 있을수 있다. 실례로 조작체계를 위한 토막등록기와 현재처리를 위한 토막등록기 등을 들수 있다.
- **첨수등록기**: 이 등록기들은 첨수주소지정에 리용되거나 자동첨수지정에 리용한다.
- **탄창지시기**: 사용자가 볼수 있는 탄창주소지정인 경우 탄창은 기억기내에 있게 되며 이 탄창의 꼭대기를 지시하는 전용등록기가 있다. 이것은 암시적인 주소지정을 할수 있게 한다. 즉 밀어넣기와 밀어꺼내기 기타 다른 탄창명령들은 탄창연산수를 명시적으로 가질 필요가 없다.

여기서도 론의해야 할 여러가지 설계문제점들이 있다. 중요한 문제점의 하나는 일반등록기들을 충분히 리용하며 또 그 리용을 전문화해야 한다는것이다. 앞의 장에서 이미 이 문제를 고찰하였다. 그것은 이 문제가 명령모임설계에 영향을 주기때문이다. 일반적으로 전문화된 등록기를 리용하게 되면 어떤 연산수지정자가 어떤 형의 등록기에 귀착되는가를 조작코드내에서 암시적으로 보여 줄수 있다. 연산수지정자는 모든 등록기들중의 어느 하나를 규정하는것이 아니라 전문화된 등록기묶음중의 어느 하나를 식별하면 되므로 따라서 비트가 절약된다. 다른 한편 이 전문화는 프로그람작성자의 유연성을 제한한다. 이 설계문제점에 대하여서는 최종적이며 가장 좋은 해결방도가 없다. 그러나 이미 언급한바와 같이 전문화된 등록기들을 리용하는것이 하나의 추세로 되고 있다.

다른 하나의 설계문제점은 일반등록기, 자료등록, 주소등록기 등의 개수이다. 이것 역시 등록기가 많을수록 더 많은 연산수지정자비트를 요구하므로 명령모임설계에 영향을 준다. 이 문제에 대해서도 이미 론의한것처럼 8~32 사이의 값을 선택하면 필요한 등록기수의 최적값을 얻을수 있다[LUND77]. 등록기수가 작으면 작을수록 기억기참조가 보다 많이 일어 나며 등록기수가 많아 진다고 하여 기억기참조가 현저히 줄어 드는것은 아니다([WILL90] 참고). 그러나 수백개의 등록기를 리용하여 특성을 개선하려는 방법은 RISC 체계에서 받아 들인것으로서 이에 대해서는 제 12 장에서 고찰한다.

마지막설계문제점은 등록기길이에 대한것이다. 주소를 보존해야 하는 등록기들은 명백히 가장 큰 주소를 유지하는데 충분한 길이를 가지고 있어야 한다. 자료등록기는 대다수 자료형에 따르는 값을 가질수 있어야 한다. 일부 콤퓨터들에서는 두개의 린접등록기들을 2배길이의 값을 보존하는 하나의 등록기로서 리용될수 있게 한다.

적어도 부분적으로나마 사용자에게 보일수 있는 마지막부류의 등록기들은 **조건코드** (기발이라고도 한다.)를 가진다. 조건코드는 CPU 하드웨어에 의해 연산결과로 설정되는 비트들이다. 실례로 산수연산은 정수, 부수, 령 혹은 자리넘침결과를 발생시킬수 있다. 등록기 혹은 기억기에 기억되는 이러한 결과외에도 다른 하나의 조건코드가 또한 설정된다. 이 코드는 조건분기연산의 한 부분으로서 그다음에 검사된다.

조건코드비트들은 하나이상의 등록기들에 집합된다. 보통 이것들은 조종등록기의 한 부분을 이룬다. 일반적으로 기계명령들은 이 비트들을 암시적인 참조로 읽는다. 프로그람작성자는 이 비트들을 변화시킬수 없다.

일부 콤퓨터에서 보조루틴호출은 사용자가 볼수 있는 모든 등록기들의 자동적인 보관을 진행하는데 이것은 복귀에서 그것을 회복하여 쓰기 위해서이다. 보관과 회복은 호출과 복귀명령의 한 부분으로서 CPU 에 의하여 수행된다. 이것은 매 보조루틴들이 사용자가 볼수 있는 등록기들을 독립적으로 리용할수 있게 한다. 그러나 일부 콤퓨터들에서는 이와는 달리 보조루틴을 호출하기에 앞서 관계되는 사용자가 볼수 있는 등록기들의 내용을 대피시키는것이 프로그람작성자가 해야 할 일로 되고 있는것도 있다. 즉 프로그람내에 대피를 위한 명령이 포함되게 된다.

2. 조종 및 상태등록기

CPU의 연산을 조종하는데 리용되는 CPU 의 등록기에는 여러가지가 있다. 대부분 콤퓨터들에서 이 등록기들은 사용자가 볼수 없다. 이 등록기들중 일부만이 조종 혹은 조작체계방식에서 집행되는 기계명령에서 볼수 있다.

물론 각이한 콤퓨터들은 서로 다른 등록기조직을 가지며 또 각이한 전문용어를 리용한다. 여기서는 간단한 서술과 함께 합리적이며 완전한 등록기형의 목록을 보기로 한다.

명령실행에서는 다음의 네가지 등록기가 기본이다.

- **프로그람계수기 (PC)**: 불러 들일 명령의 주소를 가진다.
- **명령등록기 (IR)**: 제일 마지막으로 불러 들인 명령을 가진다.
- **기억기주소등록기 (MAR)**: 기억기에서 그 위치주소를 가지다.
- **기억기완충등록기 (MBR)**: 기억기에 씌여 질 자료단어 혹은 제일 마지막으로 읽은 단어를 가진다.

PC 는 늘 실행할 다음명령을 지시하도록 매 명령을 불러 들인후 CPU 에 의하여 그 내용을 갱신한다. 분기 혹은 건너뛰기명령에 의해서도 PC 내용이 수정될수 있다. 불러 들인 명령은 IR 에 넣어 져 여기에서 조작코드와 연산수지정자가 해석된다. 자료는 MAR 와 MBR 를 리용하여 기억기에서 교환된다. 모선-구성체계에서 MAR 는 주소모선에 직접 련결되며 MBR 는 자료모선에 직접 련결된다. 사용자가 볼수 있는 등록기들은 차례로 MBR 와 자료를 교환한다.

우에서 언급된 4 개의 등록기들은 CPU 와 기억기사이 자료이동에 리용된다. CPU 내에서 자료는 처리를 위하여 ALU 에 들어 간다. ALU 는 MBR 와 사용자가 볼수 있는 등록기들에 직접 호출할수 있다. ALU 와의 경계에는 추가적인 완충등록기들이 있는데 이 등록기들은 ALU 의 입구 혹은 출구로 작용한다. 또한 이 등록기들은 MBR 와 사용자가 볼

수 있는 등록기들과의 자료교환에도 리용한다.

모든 CPU 설계에서는 **프로그람상태단어**(PSW)로 잘 알려 진 등록기 혹은 등록기묶음설계가 동반된다. 이 PSW는 조건코드와 기타 상태정보를 가진다. PSW의 공동마당 혹은 기발들은 다음과 같다.

- **부호**: 마지막산수연산결과의 부호비트를 가진다.
- **령**: 결과가 0일 때 설정된다.
- **자리올림**: 연산결과 자리올림(더하기)이 생기거나 자리내림(덜기)이 생겼을 때 설정된다. 여러단어산수연산에 리용한다.
- **같다**: 론리비교결과가 같을 때 설정된다.
- **자리넘침**: 산수연산의 자리넘침을 지시하는데 리용된다.
- **새치기가능/불가능**: 새치기를 가능 혹은 불가능하게 하는데 리용된다.
- **체계관리자**: CPU가 체계관리자 혹은 사용자방식으로 동작하고 있는가를 가리킨다. 일부 특권명령들은 체계관리자방식에서만 집행되며 어떤 기억기령역은 체계관리자방식에서만 호출할수 있다.

CPU설계에서는 조종 및 상태와 관련한 여러개의 다른 등록기설계도 진행한다. PSW외에도 추가적인 상태정보를 가지는 기억기블로크(처리조종블로크)에 대한 지시기가 있다. 벡토르화된 새치기를 리용하는 콤퓨터들에서는 새치기벡토르등록기를 장비하고 있다. 탄창을 일정한 기능(보조루틴호출)을 실현하는데 리용하는 경우 체계탄창지시기가 리용된다. 페지표지시기는 가상기억기체계에 리용된다. 마지막으로 등록기들은 I/O연산의 조종에도 리용할수 있다.

여러개의 인자들이 조종 및 상태등록기조직의 설계에 관여한다. 한가지 기본문제는 조작체계지원이다. 보통 일정한 형의 조종정보는 조작체계에서의 특별한 응용을 위해서이다. CPU 설계가가 조작체계의 기본적인것을 리해하고 있다면 등록기조직을 어느 정도까지는 조작체계에 맞출수 있다.

다른 한가지 기본설계문제는 등록기와 기억기사이 조종정보의 할당이다. 기억기의 가장 낮은 위치의 100∼1000단어정도는 조종목적에 전용으로 리용되는것이 일반적이다. 설계가는 얼마나 많은 조종정보를 등록기나 기억기에 두어야 하는가를 판단하여야 한다. 결국 비용 대 속도의 합리적인 방도를 찾아야 한다.

3. 극소형처리장치의 등록기조직에 대한 실례

여러 체계들의 등록기조직에 대한 고찰과 함께 그 호상비교는 등록기조직을 리해하는데서 의의를 가진다. 이 절에서는 대략 같은 시기에 설계된 2개의 16bit 극소형처리장치인 Motorola MC68000[STRI79]과 Intel 8086[MORS78]에 대하여 고찰한다. 그림 11-3ㄱ와 11-3ㄴ는 이 처리장치들의 등록기조직을 보여 주고 있다. 그림에서는 기억기주소등록기와 같은 순수 내부등록기들은 보여 주지 않았다.

MC68000은 32bit의 등록기들을 8개의 자료등록기와 9개의 주소등록기로 나눈다. 8개의 자료등록기는 기본적으로 자료조작에 리용되며 또 첨수등록기로서 주소지정에 리용되기도 한다. 등록기들의 너비는 8, 16 혹은 32bit 의 자료연산을 할수 있게 되여 있으며 이것은 조작코드에 의해 결정된다. 주소등록기는 32bit의 주소를 가진다. 주소등록기들중 2개는 탄창지시기로 리용된다. 두개의 탄창지시기에서 하나는 사용자용으로, 다른 하나는 조작체계용으로 리용된다. 이 등록기들에는 7이라는 번호가 붙어 있는데 그것은 한번에 하나만을 리용할수 있기때문이다. MC68000 은 또한 32bit 의 프로그람계수기와 16bit

의 상태등록기를 가지고 있다.

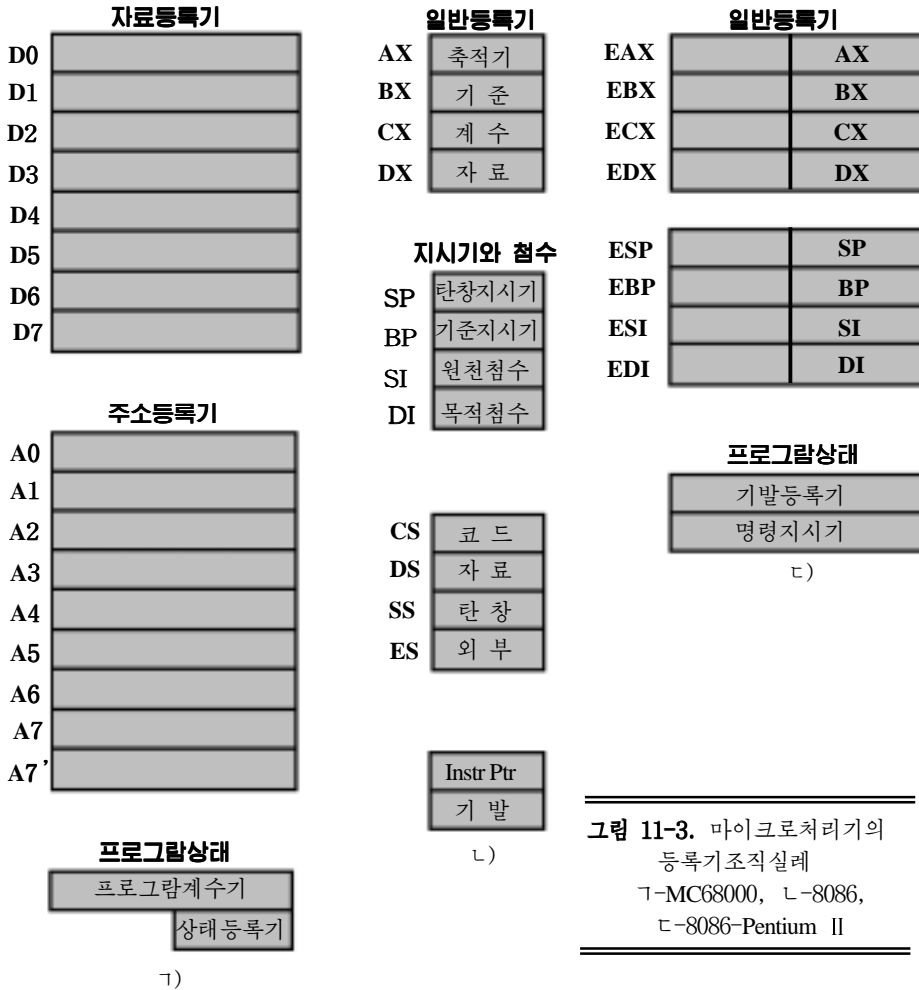



그림 11-3. 마이크로처리기의 등록기조직실례
ㄱ-MC68000, ㄴ-8086,
ㄷ-8086-Pentium II

Motorola 연구집단은 특별한 목적을 위한 등록기들을 쓰지 않는 매우 규칙적인 명령모임을 설계하는것을 목표로 하였다. 이러한 코드효과성에 대한 관심은 등록기들을 2 개의 기본구성요소들로 분할할수 있게 하였다. 이렇게 하면 매 등록기지정자에서 한비트를 절약하게 된다. 이것은 코드의 일반성을 보장하면서도 코드를 함축할수 있는 합리적인 방안으로 된다.

Intel 8086 은 등록기조직에서 여러가지 방법을 리용하고 있다. 여기서는 일부 등록기들이 일반목적에 리용되고 있다고 해도 모든 등록기들을 특별한 목적 즉 전용으로 리용한다. 8086 은 1byte 혹은 16bit 단위로 주소를 지정할수 있는 4 개의 16bit 자료등록기와 4 개의 16bit 지시 및 첨수등록기를 가지고 있다. 자료등록기들은 일부 명령들에서 일반목적에 리용된다. 기타 다른 등록기들은 암시적으로 리용된다. 실례로 곱하기명령은 늘 축적기를 리용한다. 4 개의 지시등록기들은 여러가지 연산에 암시적으로 리용되며 이 연산은 토막변위를 가진다. 8086 에는 4 개의 16bit 토막등록기들이 있다. 4 개의 토막등록기들중 3 개는 전용으로 리용되며 암시적으로 현재명령토막 (분기명령에 효과적이다.), 자료토막, 탄창토

막을 지시하는데 리용된다. 전용 및 암시적인 코드의 리용은 유연성을 줄이지만 코드를 압축할수 있게 한다. 8086 은 또한 명령지시기, 한비트상태 및 조종기발들의 묶음을 가지고 있다.

이상에서 본 두 처리장치에서의 등록기조직에 대한 고찰은 CPU 등록기들을 구성하는 가장 좋은 방법으로 되는 기본원리가 세계적으로 아직 없다는것을 보여 주고 있다 [TOON81]. 전체 명령모임설계와 많은 다른 CPU 설계론쟁점들에서 본바와 같이 이것들은 어디까지나 판단과 맛을 본데 지나지 않는다.

등록기조직설계와 관련한 두번째 교훈적인 점은 그림 11-3 ㄷ에서 설명하고 있다. 이 그림은 사용자가 볼수 있는 등록기조직을 Intel 80386 에 대하여 보여 주고 있다 [ELAY85]. 80386 은 8086 을 확장하여 설계한 32bit 극소형처리장치이다. 따라서 80386 은 32bit 등록기들을 가지고 있다.[1] 80386 에서는 그 이전의 콤퓨터들에서 작성된 프로그람들과의 호환성을 보장하기 위하여 새로운 구성을 보충하는것과 함께 종래의 등록기조직을 유지하고 있다. 결국 이러한 설계원리에 기초하고 있으므로 32bit 처리장치기본방식 역시 등록기조직을 설계하는데서 유연성이 제한되고 있다.

제 3 절. 명령주기

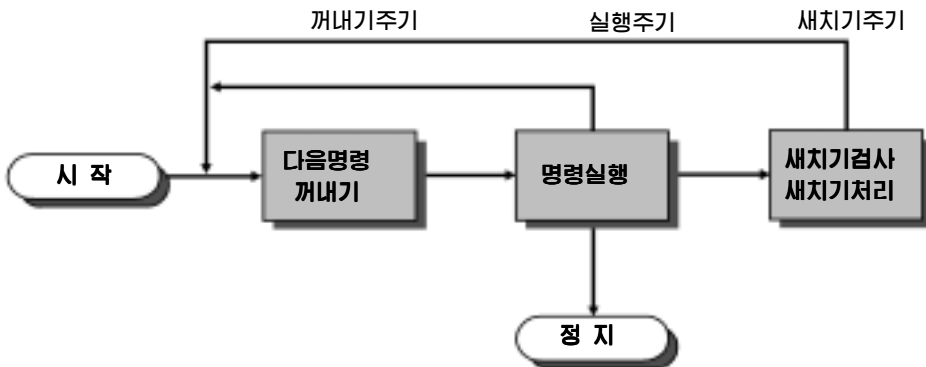



그림 11-4. 새치기를 가진 명령주기

이미 제 3 장 제 2 절에서 명령주기를 고찰하였다. 그림 11-4 는 제 3 장 제 2 절에서 명령주기를 설명하는 그림들중의 하나(그림 3-9)를 되풀이한것이다. 앞에서 고찰한것을 상기해 보면 명령주기는 다음과 같은 부분주기로 이루어 진다.

- **꺼내기** : 기억기에서 CPU 로 다음명령을 읽는다.
- **실행** : 조작코드를 해석하여 지정된 연산을 수행한다.
- **새치기** : 새치기가 가능하여 새치기가 일어 나면 현재처리상태를 대피시키고 새치기처리를 한다.

여기서는 명령주기에 대한 보다 상세한 고찰을 진행한다. 이를 위하여 먼저 간접주기라고 하는 한가지 보충적인 부분주기를 고찰한다.

---

[1] MC68000 이 이미 32bit 등록기를 리용하고 있으므로 이것의 완전한 확장인 MC68020[MACG84]은 같은 등록기조직을 리용한다.

1. 간접주기

이미 제 10 장에서 명령실행은 기억기호출을 요구하는 하나이상의 연산수를 동반한다는것을 보았다. 더우기 간접주소지정방식이 리용되면 추가적인 기억기호출이 요구된다.

간접주소의 꺼내기는 하나이상의 명령부분주기에서 일어 난다고 볼수 있다. 이 결과를 보여 준것이 그림 11-5 이다. 이 그림에서 기본은 명령꺼내기와 명령실행의 호상교차부분이다. 명령이 CPU 로 들어 오면 임의의 간접주소지정방식이 동반되는가를 검사한다. 동반된다면 필요한 연산수를 간접주소지정방식을 리용하여 불러 들인다. 그다음 실행이 뒤따르며 다음명령을 꺼내기전에 새치기가 있으면 그 처리를 한다.

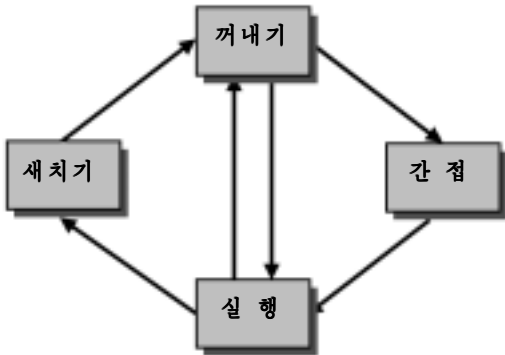



그림 11-5. 명령주기

이 처리에 대한 다른 한가지 고찰방법을 그림 11-6 에 보여 주었다. 이 그림은 그림 3-12 를 수정한것이다. 그림은 명령주기의 본질을 보다 더 정확히 설명하고 있다. 명령을 불러 들이면 곧 그의 연산수가 식별된다. 그다음 기억기에서 매 입구연산수를 불러 내며 이 처리는 간접주소지정방식을 요구할수 있다. 등록기연산수들은 불러 들일 필요가 없다. 조작코드가 실행되면 곧 주기억기에 결과를 기억하는 처리가 진행된다.

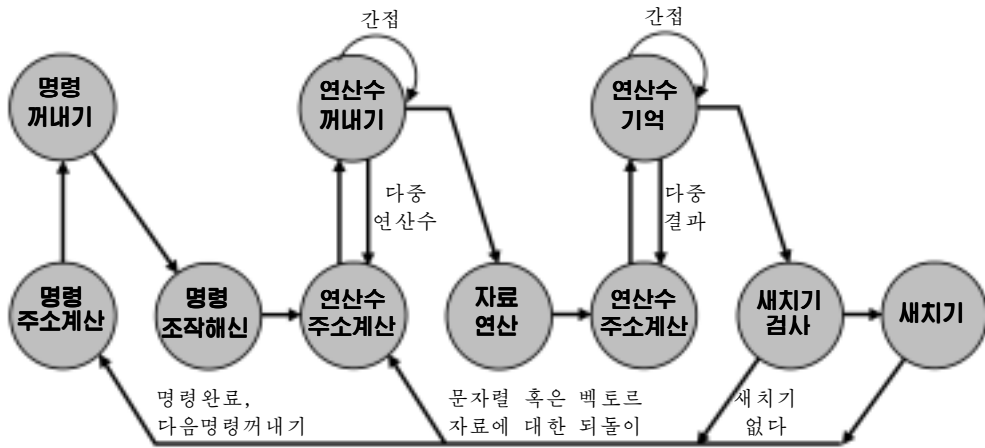



그림 11-6. 명령주기상태도

2. 자료흐름

한 명령주기동안에 실행되는 사건들의 정확한 순서는 CPU 설계에 관계된다. 그러나 무엇이 일어 나야 한다는것을 일반적인 용어로 지적할수는 있다. 이제 CPU 가 기억기주소등록기(MAR), 기억기완충등록기(MBR), 프로그람계수기(PC), 명령등록기(IR)를 가지고 있다고 하자.

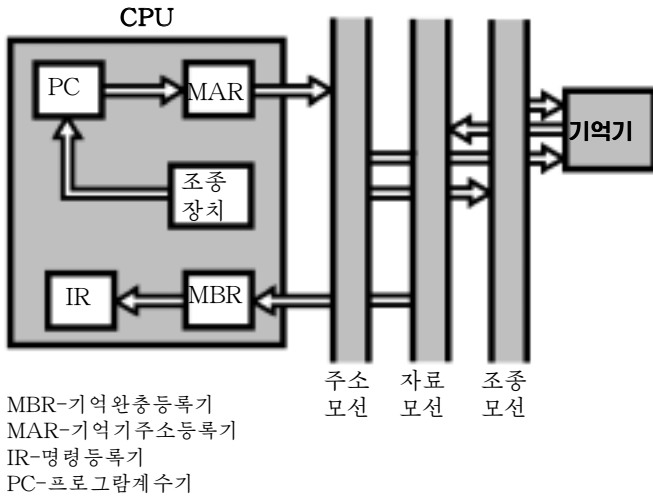



그림 11-7. 자료흐름, 꺼내기주기

꺼내기주기동안에 하나의 명령이 기억기로부터 읽어 진다. 그림 11-7 은 이 주기동안의 자료흐름을 보여 주고 있다. PC 는 불러 들여야 할 다음명령의 주소를 가진다. 이 주소는 MAR 로 이동되여 주소모선에 놓이게 된다. 조종장치는 기억기읽기를 요구하며 결과를 자료모선에 태워 MBR 에 복사하고 그다음 IR 로 옮긴다. 그 동안 PC 는 하나 증가되여 다음명령을 위하여 준비된다.

꺼내기주기가 지나자마자 조종장치는 간접주소지정을 리용하는 연산수지정자를 가지고 있는가를 판단하기 위하여 IR 등록기의 내용을 검사한다. 간접주소지정방식이라면 **간접주기**가 수행된다. 그림 11-8 에서 보여 준바와 같이 이것은 간단한 주기이다. 주소참조에 리용되는 MBR 의 제일 오른쪽 N 개의 비트들은 MAR 로 이동된다. 그다음 조종장치가 MBR 에 필요한 연산수 주소를 가지도록 기억기읽기를 요구한다.

그림 11-8. 자료흐름, 간접주기

꺼내기와 간접주기는 간단하며 예측할수 있다. 실행주기는 여러가지 기계명령들중 어느것이 IR 내에 있는가에 따라 많은 형태를 가진다. 이 주기는 등록기에로의 자료이동, 기억기 혹은 I/O 로부터의 읽기 혹은 쓰기, ALU 의 작용을 동반할수 있다.

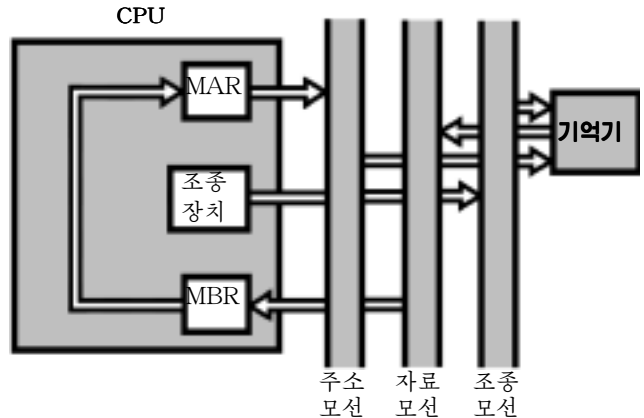
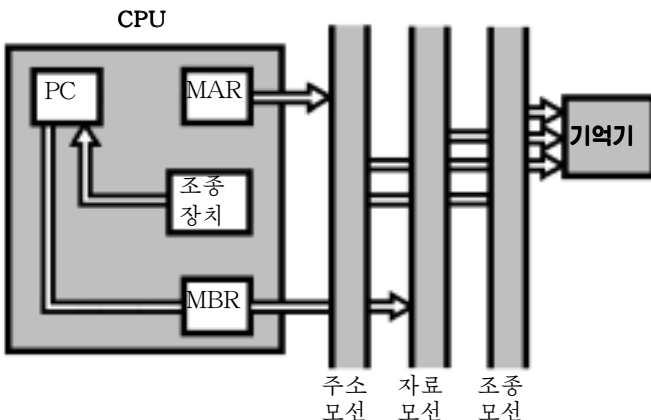



그림 11-9. 자료흐름, 새치기주기

새치기주기도 꺼내기 및 간접주기와 같이 단순하며 예측가능하다 (그림 11-9). PC 의 현재내용은 CPU 가 새치기후에 정상동작을 계속 할수 있도록 보관되여야 한다. 따라서 PC 의 내용은 기억기내의 MBR 로 이동된다. 이 목적으로

예약된 전용기억기위치주소는 조종장치로부터 MAR에 넣어 진다. 이를 위하여 탄창지시기를 리용할수 있다. PC에는 그다음 새치기루틴의 주소가 넣어 진다. 결과 적당한 명령을 불러 들여 다음명령주기를 시작한다.

제 4 절. 명령관흐름처리

콤퓨터체계가 발전하면서 보다 고속인 회로를 비롯한 선진기술의 우점들을 받아 들여 그의 성능은 크게 개선되였다. 콤퓨터의 성능개선에는 CPU 구성의 강화도 큰 영향을 미친다. 앞에서 이미 이와 관련한 실례들 즉 단일축적기보다 여러개의 등록기나 캐쉬의 리용이 더 우월하다는것을 고찰하였다. 성능개선을 위한 공통적이며 일반적인 다른 한가지 구성방법은 명령의 관흐름처리이다.

1. 관흐름처리의 전략

명령의 관흐름처리는 제조업분야에서 조립선의 리용과 류사한 개념이다. 조립선은 하나의 생산물이 여러 단계를 거쳐 나온다는 사실을 리용하고 있다. 조립선의 여기저기에서 생산처리를 하게 하면 각이한 단계에서 생산물이 동시에 나올수 있다. 이 처리를 **관흐름처리**라고 말할수 있는데 그것은 관흐름에서와 같이 이미 접수된 입구들이 다른 끌에서 출구에 나타나기도전에 새로운 입구들이 접수되기때문이다.

명령실행에 이 개념을 적용하자면 먼저 명령이 여러 단계를 가진다는것을 리해해야 한다. 실례로 그림 11-6에서 명령주기는 차례로 일어 나는 10개의 과제로까지 될수 있다. 이로부터 관흐름처리를 위한 일부 좋은 기회가 있다는것을 알수 있다.

이제 단순한 방법으로서 명령처리를 두 단계 즉 명령꺼내기단계와 명령실행단계로 나누어 고찰해 보자. 주기억기를 호출하지 않는 명령실행동안에는 시간이 있게 된다. 이 시간을 현재명령의 실행과 병렬로 다음명령을 불러 내는데 리용할수 있다. 이 관흐름은 두개의 독립적인 단계를 가진다. 즉 첫 단계에서는 명령을 불러 내여 그것을 완충(림시기억)한다. 두번째 단계가 놀고 있다면 첫 단계는 완충된 명령을 통과시킨다. 두번째 단계에 들어 가서 명령이 실행되는 동안 첫 단계는 임의의 리용되지 않고 있는 기억기주기를 리용하여 다음명령을 불러 내여 완충한다. 이것을 **명령미리꺼내기**(instruction prefetch) 혹은 **꺼내기중첩**이라고 한다.

그림 11-10. 2단계 명령흐름선
ㄱ-단순고찰, ㄴ-확장고찰

이렇게 하면 명령실행의 속도를 높일수 있다. 만일 꺼내기단계와 실행단계가 같은 시간에 있다면 명령주기시간은 절반으로 줄어들것이다. 그러나 구체적으로 관흐름을 고찰해 보면(그림 11-10 ㄴ) 실행속도가 두가지 리유로 2배로까지는 빨라 지지 않는다는것을 알수 있다.

1. 실행시간은 일반적으로 꺼내기시간보다 더 길다. 명령은 연산수의 읽기와 기억 그리고 일부 성능이 다른 연산을 동반한다. 따라서 꺼내기단계는 그의 완충기가 빌 때까지 얼마간 기다려야 한다.
2. 조건분기명령은 불러 들여야 할 다음명령의 주소를 알수 없게 한다. 따라서 꺼내기단계는 실행단계로부터 다음명령의 주소를 받을 때까지 기다려야 한다. 실행단계는 다음명령을 불러 내는 동안 기다려야 한다.

두번째 리유로부터 시간손실을 추측대로 감소시킬수 있다. 단순한 규칙은 다음과 같다. 조건분기명령이 꺼내기단계로부터 실행단계로 넘어 가면 꺼내기단계는 분기명령후에 기억기에서 다음명령을 불러 낸다. 다음명령이 분기명령이 아니면 시간손실은 없게 된다. 분기명령인 경우는 불러 낸 명령이 포기되며 다음명령을 불러 들인다.

이상에서 본 인자들이 2 단계 관흐름의 잠재적인 효과성은 감소시키지만 일부 속도상승도 가져 온다. 보다 더 높은 속도상승을 얻자면 관흐름이 보다 많은 단계를 가져야 한다. 이제 명령처리과정을 다음과 같이 분할해 보자.

- **명령꺼내기(FI)**: 다음에 실행해야 할 명령을 완충기에 넣기
- **명령해신 (DI)**: 조작코드와 연산수지정자를 해신
- **연산수계산(CO)**: 매 원천연산수의 유효주소계산. 여기에는 변위,등록기간접, 간접 기타 다른 양식의 주소계산이 동반된다.

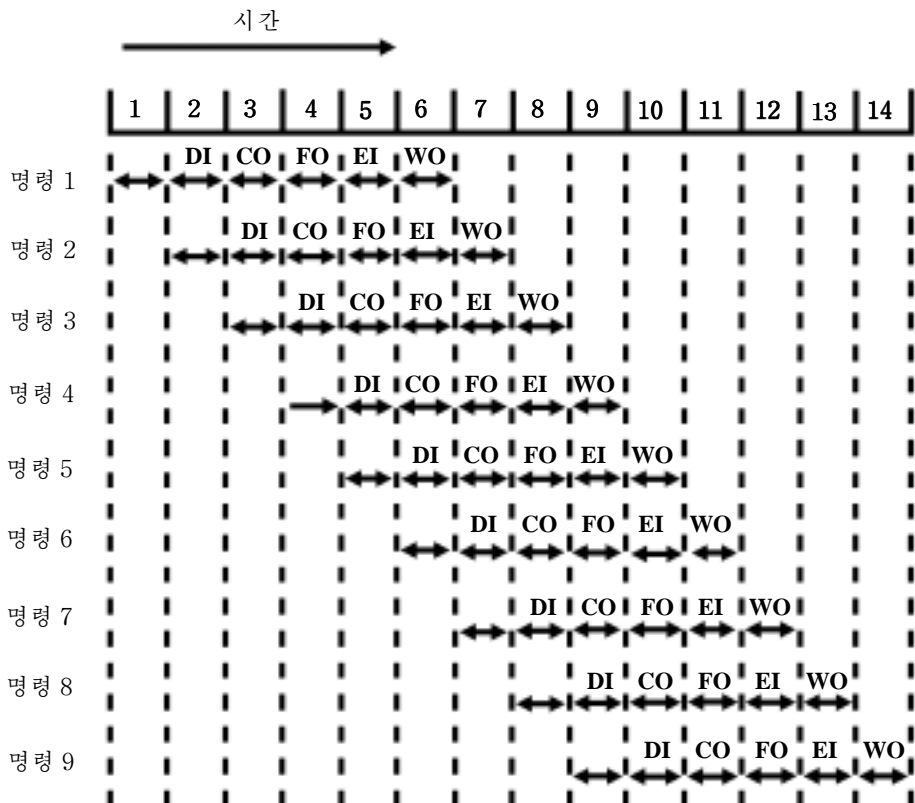



그림 11-11. 명령흐름선동작의 시간선도

- **연산수꺼내기(FO)**: 기억기로부터 매 연산수를 꺼내기. 등록기로 연산수를 불러들일 필요는 없다.
- **명령실행 (EI)**: 지정된 연산을 수행하고 결과를 지정된 목적연산수위치에 기억
- **연산수쓰기 (WO)**: 기억기에 결과를 기억

이와 같이 세분화하여 분할하면 여러개의 단계들이 보다 가깝게 같은 존속시간에 있게 된다. 설명을 위하여 이 단계들이 같은 시간에 있다고 가정한다. 그림 11-11은 6개의 단계를 가진 관흐름이 9개의 명령에 해당한 실행시간을 54시간단위로부터 14시간단위로 줄일수 있다는것을 보여 주고 있다.

여러개의 설명문들이 차례로 있다. 그림에서 선도는 매 명령이 관흐름의 모든 6개의 단계들을 통과한다고 가정하고 그린것이다. 그러나 늘 이와 같이 되지는 않는다. 실례로 넣기명령은 W0단계가 필요 없다. 그러나 관흐름하드웨어를 간단히 하기 위하여 매 명령이 6 개의 모든 단계를 요구한다고 가정하고 시간선도를 그렸다. 또한 선도는 모든 단계들이 병렬로 수행된다고 가정하고 그렸다. 그리고 특별히 기억기충돌은 없다고 가정한다. 실례로 FI, FO와 W 단계들은 기억기호출을 동반하는데 시간선도는 이 호출들이 동시에 일어 날수 있다는것을 암시하고 있다. 그러나 대다수 기억기체계에서는 이것이 허용되지 않는다. 그렇지만 필요한 값이 캐쉬내에 있을수도 있고 FO 혹은 W0단계가 없을수도 있다. 따라서 기억기충돌은 많은 시간동안 관흐름을 지체시키지는 않는다

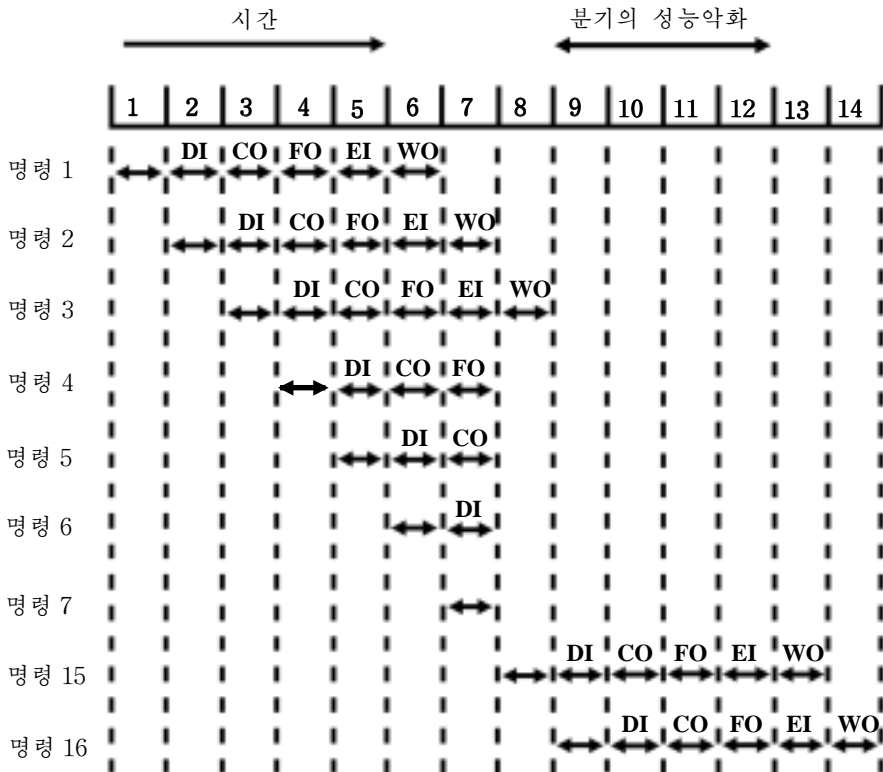



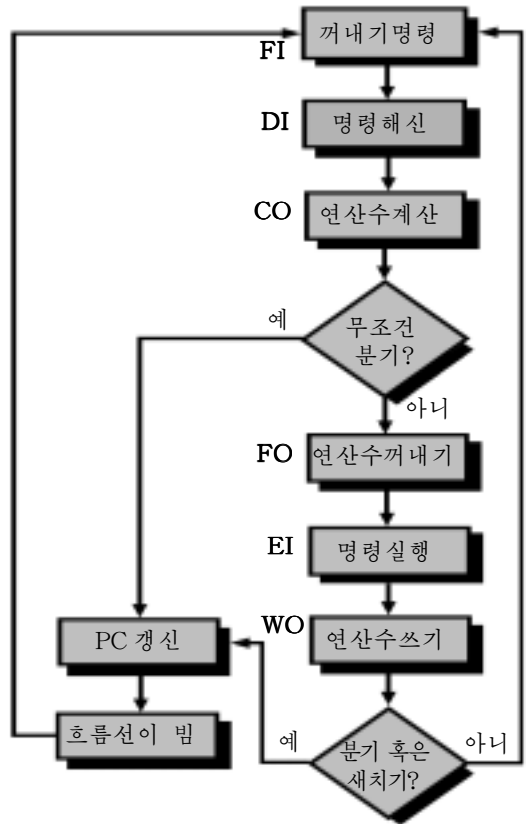
그림 11-12. 명령흐름선동작에 미치는 조건분기의 영향

성능을 높이는데는 이외에도 여러가지 다른 인자들이 영향을 준다. 만일 6개의 단계들이 같은 존속기간에 있지 않다면 두 단계의 관흐름에서 이미 고찰한바와 같이 여러개의 관흐름단계에서 동반되는 일부 기다림이 있게 된다. 성능제고에 저애를 주는 다른 하나의 인자는 조건분기명령이다. 여기서 제일 중요한것은 새치기의 발생이며 이 발생은 예측불가능한 사건으로 된다. 그림 11-12에 그림 11-11과 같은 프로그람의 리용에서 조건분기의 영향을 보여 주었다. 그림에서는 명령 3이 명령 15에 대한 조건분기라고 가정하고 있다. 명령이 실행될 때까지는 어느 명령이 다음에 올것인가를 알수 없다. 이 실례에서 관흐름은 단순히 차례로 다음명령(명령 4)을 넣고 처리한다. 그림 11-11에서는 분기가 없으며 완전히 성능이 제고된다. 그러나 그림 11-12에는 분기가 있으며 이것은 시간단위 7이 끝날 때까지 확정되지 않는다. 이 시점에서 관흐름은 쓸모 없는 명령들을 지운다. 시간단위 8기간에는 명령 15가 관흐름에 들어 온다. 명령은 시간단위 9로부터 시간단위 12사이에 완전히 실행되지 못한다. 이것은 분기를 예견하지 못한것으로 하여 초래되는 성능악화이다. 그림 11-13에는 분기 및 새치기를 고려하여 관흐름처리에 필요한 론리흐름을 보여 주었다.

여기에서는 앞에서 고찰한 단순한 두 단계구성에서는 나타나지 않았던 문제들이 발생한다. CO단계는 아직도 관흐름에 남아있는 앞선 명령에 의하여 교체될수 있는 등록기내용에 관계된다. 이때 다른 등록기와 기억기와의 충돌이 일어 날수 있다. 따라서 체계는 이와 같은 충돌을 고려한 론리에 따라 동작해야 한다.

앞에서 이미 고찰한 론의로부터 관흐름에 더 많은 단계들이 있으면 있을수록 실행속도가 더 빨라 진다는것을 알수 있다. 일부 IBM S/360의 설계가들은 고성능설계에서 이러한 매우 단순한 현상에 저애를 주는 다음과 같은 두가지 요인들을 지적하였다[ANDE 67].

그림 11-13. 6개 단계의 CPU 명령흐름선

- 관흐름의 매 단계에는 완충기에서 완충기로 자료를 이동하거나 여러가지 준비 및 전달기능을 수행하는데 걸리는 간접소비시간이 있다. 이 시간은 단일명령의 총체적인 실행시간을 얼마간 길게 할수 있다. 이것은 련속되는 명령들이 빈번한 분기의 리용이나 기억기호출의 의존성을 통하여 론리적으로 종속되는 경우에 의미를 가진다.
- 기억기 및 등록기의 의존성을 조종하며 관흐름리용을 최적화하기 위하여 필요한 조종론리의 량은 단계수가 많아 지면 상당히 증가한다. 이것은 단계들사이 이행을 조종하는 론리가 단계들을 조종하는것보다 더 복잡하다는것을 말해 준다.

명령관흐름처리는 성능을 제고하기 위한 위력한 기술이지만 복잡성을 합리적으로 구성하여 최적인 결과를 얻자면 설계를 주의하여야 한다.

2. 관흐름의 성능

여기서는 관흐름의 성능과 상대적인 속도상승에 대한 정량적인 고찰을 간단히 한다([HWAN93]에서의 론의에 기초하고 있다.). 명령관흐름의 주기시간 τ는 관흐름을 통하여 명령 한 단계의 모임을 만드는데 필요한 시간이다. 그림 11-11 과 11-12 에서 매렬은 한주기시간을 표시한다. 이 주기시간은 다음과 같이 결정된다.

$$\tau = \max[\tau_i] + d = \tau_m + d \qquad i, 1 \leqq i \leqq k$$

여기서

τ_m-최대단계지연(가장 큰 지연을 주는 단계의 지연),
k -명령관흐름에서 단계들의 수,
d -한 단계에서 다른 단계로 신호나 자료를 전파시키는데 요구되는 빗장의 시간지연

일반적으로 시간지연 d는 박자임풀스와 같으며 $\tau_m \gg d$의 관계를 가진다. 이제 n개의 명령이 분기가 없이 처리된다고 하자. n개의 모든 명령을 실행하는데 요구되는 총 시간 T_k는 다음과 같다.

$$T_k = [k + (n - 1)]\tau \qquad (11\text{-}1)$$

총 k 개의 주기는 첫번째 명령의 실행을 완성하는데 걸리는 시간이며 나머지 n-1 개의 명령들은 n-1 개의 주기를 요구한다.[2] 이 식은 그림 11-11 로부터 쉽게 확증된다. 아홉번째 명령은 시간주기 14 에서 완성된다.

$$14 = [6 + (9 - 1)]$$

명령관흐름에서 속도상승인자는 관흐름이 없는 실행과 비교한 경우 다음과 같다.

$$S_k = \frac{T_1}{T_k} = \frac{nk\tau}{[k+(n-1)]\tau} = \frac{nk}{k+(n-1)} \qquad (11\text{-}2)$$

그림 11-14 ㄱ 는 분기가 없이 실행되는 명령개수의 함수로서 속도상승인자를 보여준다. 기대했던대로 극한(n→∞)에서 k배의 속도상승이 일어 난다. 그림 11-14 ㄴ는 명령관흐름에서 단계수의 함수인 속도상승인자를 보여 준다.[3] 이 경우에 속도상승인자는 분기가 없이 관흐름에 공급될수 있는 명령의 개수로 다가간다. 따라서 관흐름단계를 많이 할수록 속도상승가능성이 더 커진다는것을 알수 있다. 그러나 실천적인 문제로서 추가적인 관흐름단계들의 도입은 비용증가, 단들사이 지연, 관흐름의 범람이 분기들에 의해서 초래될것이라는 사실 등을 낳는다. 결국 임의의 관흐름에서 그 단계의 수가 6~9 단계를 넘어 서도록 추가적으로 단계를 늘이면 비생산적이라는것을 알수 있다.

---

[2] 주기시간은 모든 단계가 다 있는 경우에만 τ의 최대값과 같아 진다. 첫 하나 혹은 몇주기동안에는 τ의 최대값보다 작게 된다.

[3] x축은 그림 11-14 ㄱ에서는 로그척도, 그림 11-14 ㄴ에서는 선형척도이다.

3. 분기처리

명령관흐름을 설계하는데서 중요한 문제의 하나는 관흐름의 초기단계들에서 명령들의 련속흐름을 보장하는것이다. 이 문제에서 기본장애로 되는것은 이미 보았지만 조건분기명령이다. 조건분기명령인 경우 명령이 실지로 실행될 때까지는 분기가 있는지 없는지 판정하기 힘들다.

그리하여 조건분기처리를 위한 다음과 같은 여러가지 방법들이 연구되였다.

- 다중흐름
- 분기목표미리꺼내기
- 고리완충기
- 분기예측
- 지연분기

그림 11-14. 명령관흐름조종의 속도상승

다중흐름

단순한 관흐름은 다음에 불러내는 두개의 명령중 어느 하나를 선택해야 하며 또 선택을 잘못할수도 있으므로 분기명령으로 인하여 성능이 악화된다. 이것을 해소하기 위한 강력한 방법으로서 관흐름의 초기부분을 사본화해놓고 두개의 흐름을 리용하도록 관흐름이 명령을 둘 다 불러 내게 하는것이다. 이 방법은 두가지 문제점을 낳는다.

- 다중관흐름인 경우는 등록기 및 기억기에 대한 호출에서 경쟁으로 인한 지연이 생긴다.
- 추가적인 분기명령이 초기분기판정이 이루어 지기도전에 관흐름(다른 흐름)에 들어 올수 있다. 이때 매 명령은 추가적인 흐름을 요구한다.

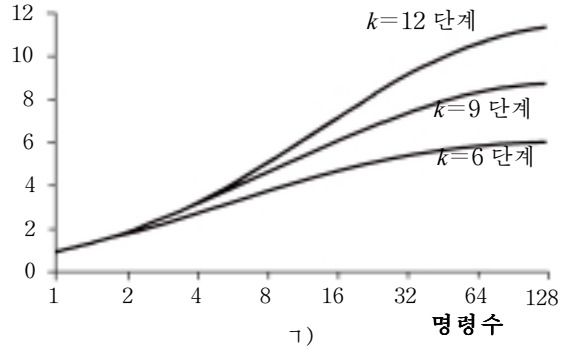
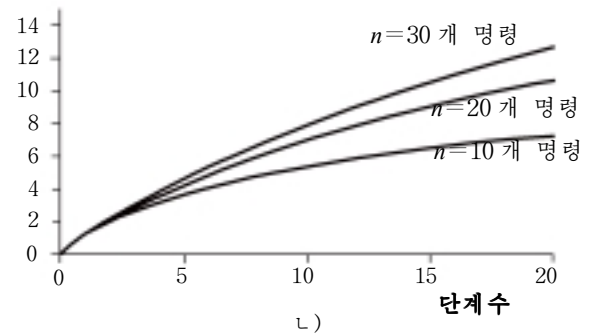
이와 같은 결함이 있음에도 불구하고 이 전략은 성능을 개선할수 있다. IBM 370/168 과 IBM3033 은 2 개 혹은 그이상의 관흐름을 가진 대표적인 콤퓨터들이다.

분기목표의 미리꺼내기

조건분기로 판정되면 분기에 뒤따르는 명령외에 분기목표를 미리 불러 낸다. 이 목표는 분기명령이 실행되는 기간 대피된다. 분기가 일어 났다면 목표는 이미 미리 불러들인것으로 된다.

IBM360/91 은 이 방법을 리용하였다.

고리완충기

고리완충기는 관흐름의 명령꺼내기단계에 리용되는 작고 속도가 매우 빠른 기억기로

서 련속적으로 불러 들인 N 개의 명령들을 기억한다. 분기가 일어 나면 하드웨어는 우선 분기목표가 이 완충기내에 있는가를 검사한다. 고리완충기내에 있다면 그 완충기로부터 불러 낸다. 고리완충기는 다음과 같은 세가지 우점을 가진다.

- 미리꺼내기를 진행하면 고리완충기는 현재명령의 꺼내기주소보다 앞에 있는 련속적인 일부 명령들을 가질수 있다. 련속적으로 불러 낸 명령들은 보통의 기억기호출시간을 들이지 않고도 리용할수 있다.
- 분기가 분기명령의 주소에서 얼마간 앞위치에서 일어 난다면 분기목표는 이미 완충기내에 있게 된다. 이것은 IF-THEN 과 IF-THEN-ELSE 명령렬의 공동발생에 쓸모가 있다.
- 이 방법은 고리 혹은 반복처리에 특별히 잘 맞는다. 따라서 이름도 **고리완충기**라고 부른다. 고리완충기가 고리내의 모든 명령을 가질수 있도록 충분히 크게 되면 이 명령들은 첫 반복순환을 위하여 기억기로부터 한번만 불러 내면 된다.

련속반복순환인 경우는 필요한 모든 명령들이 이미 완충기내에 있게 된다. 고리완충기는 원리적으로는 명령들에 전용으로 리용되는 캐쉬와 류사하다. 차이점은 고리완충기가 오직 련속명령들만을 가지며 크기가 훨씬 더 작고 따라서 비용이 눅다는것이다.

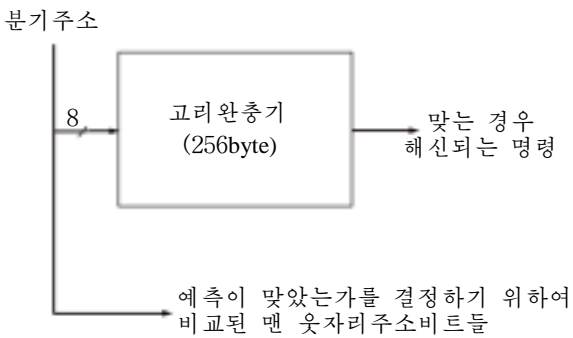



그림 11-15. 고리완충기

그림 11-15 는 고리완충기의 한가지 실례를 보여 주고 있다. 완충기가 256byte 를 가지고 바이트주소지정을 리용한다면 맨 아래자리 8 개의 비트들이 완충기를 지정하는데 리용된다. 나머지 대다수 비트들은 분기목표가 완충기라는 포위된 환경내에 있는가를 판정하도록 검사된다.

고리완충기를 리용하는 콤퓨터들중에는 일부 CDC 콤퓨터들(Star-100, 6600, 7600)과 CRAY-1 이 있다. Motorola 68010 에서는 전문화된 형식의 고리완충기를 리용하고 있는데 이 고리완충기는 DBcc(조건에 대한 감소 및 분기)명령(이 장의 련습문제 6 참고)을 포함하는 3 개의 명령으로 된 고리를 가질수 있다. 3 개의 단어완충기는 명령들을 보유하여 고리조건이 만족되는 기간은 처리장치가 이 명령들을 반복하여 실행할수 있게 한다.

분기예측

분기가 일어 나겠는가를 예측하는데는 여러가지 방법들이 있다. 그중에서 일반적인 것들은 다음과 같다.

- 절대로 일어 날수 없다고 예측
- 늘 일어 난다고 예측
- 조작코드에 의한 예측
- 일어 남/일어 나지 않음스위치
- 분기경력표

우의 다섯가지 방법들가운데서 첫 세개의 방법들은 정적방법들이다. 즉 이 방법들은

조건분기명령시간까지의 실행경력에 의존하지 않는다. 마지막 두 방법은 동적방법으로서 실행경력에 의존한다.

다섯가지 방법들가운데서 첫 두 방법이 가장 단순하다. 이 방법들은 늘 분기가 일어나지 않으므로 련속적으로 명령을 불러 내는것을 계속하거나 늘 분기가 일어 나므로 분기목표로부터 불러 낸다고 가정한다. 68020 과 VAX 11/780 은 이외에도 잘못된 판정의 영향을 최소로 하기 위한 특징도 포함하고 있다. 만일 분기후의 명령꺼내기가 페지결함 혹은 보호위반을 일으킨다면 처리장치는 명령을 불러 내야 한다는것이 명백해 질 때까지 그의 미리꺼내기동작을 중단한다.

프로그람동작에 대한 분석결과는 조건분기가 그 시간의 50 %보다는 더 자주 일어난다는것을 보여 주고 있다[LILJ88]. 그리고 두 경로로부터의 꺼내기비용이 같은 경우에는 늘 분기목표주소로부터의 미리꺼내기가 뒤따르는 경로로부터의 미리꺼내기보다는 더 성능이 좋다는것을 보여 주고 있다. 그러나 페지화된 콤퓨터들에서는 분기목표를 미리 불러 내는것이 다음명령을 련속적으로 미리 불러 내는것보다 페지결함을 더 쉽게 가져오므로 이 성능약화를 고려해야 한다. 성능회복기구는 이 악화를 감소시킬수 있도록 적용되여야 한다.

정적방법들가운데서 제일 마지막방법은 분기명령의 조작코드에 기초하여 판정을 진행한다. 처리장치는 분기가 일정한 조작코드에 대하여서만 일어 나며 기타는 일어 나지 않는다고 본다. 문헌 [LILJ88]은 이 전략을 리용하여 75 %이상의 성공률을 얻을수 있다는것을 보여 주고 있다.

동적인 분기전략은 프로그람내에서 조건분기명령의 경력을 기록하는것으로서 예측정확도를 높이고 있다. 실례로 하나이상의 비트들은 명령의 최신경력을 되돌리는 매 조건분기명령과 련관시킬수 있다. 이 비트들을 일어 남/일어 나지 않음스위치라고 한다. 이 스위치는 명령을 만나는 다음시간에 알맞는 판정을 진행하도록 처리장치에 지시하는 역할을 한다. 이 경력비트들은 주기억기내의 명령과는 련관이 없다. 오히려 이 비트들은 림시고속기억기내에 보존된다. 이 비트들을 캐쉬내에 있는 임의의 조건분기명령과 련관시킬수 있는 가능성이 있다. 명령이 캐쉬내에 재배치될 때 그의 경력은 없어 진다. 다른 또 한가지 가능성은 캐쉬에로의 기입시에 하나 혹은 그이상의 비트들에 최근에 실행된 분기명령을 위한 작은 표를 보존하는것이다. 처리장치는 캐쉬와 같이 련상적으로 표를 호출하거나 혹은 분기명령주소의 낮은 자리비트들을 리용하여 호출할수 있다.

단일비트를 리용하는 경우 기록될수 있는것은 이 명령의 마지막실행이 분기로 끝나는가 아니면 분기로 끝나지 않는가 하는것뿐이다. 단일비트를 리용할 때의 결함은 고리명령과 같이 거의 늘 일어 날수 있는 조건분기명령인 경우에 나타난다. 한비트로 된 경력을 리용하면 오유가 고리를 리용할 때마다 두번 발생한다. 즉 고리에 들어 가면서 한번, 고리에서 빠져 나오면서 한번 발생한다.

두개의 비트를 쓰는 경우는 관련된 명령실행의 마지막 두 경우의 결과를 기록하거나 일부 다른 형식에서의 상태를 기록하는데 리용할수 있다. 그림 11-16 은 이와 관련한 전형적인 방법을 보여 주고 있다(다른 가능성을 위하여 이 장의 련습문제 5 참고). 판정처리는 4 개의 상태를 가진 유한상태기계로 표시할수 있다. 주어 진 명령의 마지막 두 분기가 같은 경로에서 일어 난다면 예측은 그 경로를 취하는것으로 된다. 그 예측이 맞지 않으면 명령과 만나는 다음시간에 여전히 같은 경로를 취한다. 예측이 또 맞지 않았다면 다음예측은 반대경로를 선택한다. 결국 알고리듬은 두번의 예측이 련속적으로 틀릴수 있다는 전제하에서 예측판정을 한다. 례를 들어 분기가 한 고리내에서 한번만 비정상적인 방향으로 일어 나면 그때 예측은 단 한번 틀릴것이다.

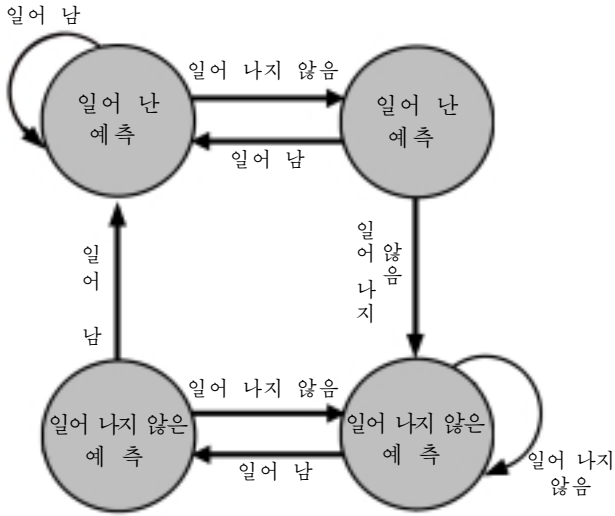



그림 11-16. 분기예측상태도

일어 남/일어 나지 않음스위치 방법을 리용하는 체계의 대표적인 실례는 IBM 3090/400 이다.

우에서 언급한바와 같이 경력비트들의 리용은 한가지 결함을 가지고 있다. 즉 판정결과가 분기가 있는것으로 되면 조건분기명령에서 연산수인 목표주소가 해신될 때까지 목표명령을 불러 낼수 없다. 분기판정이 진행되자 곧 명령꺼내기를 시작할수 있으면 보다 큰 효과성을 기대할수 있다. 이를 위해서는 보다 많은 정보를 분기목표완충기 혹은 분기경력표에 가지고 있어야 한다.

분기경력표는 관흐름의 명령꺼내기단계와 련관되는 작은 캐쉬이다. 표에로의 기입은 세가지 요소 즉 분기명령의 주소, 그 명령의 리용상태를 기록하는 몇개의 경력비트들의 개수, 목표명령에 대한 정보로 이루어 진다. 대다수 제안들과 그 실현에서는 이 세번째 마당이 목표명령의 주소로 된다. 세번째 마당에는 목표명령도 포함시킬수 있다. 세번째 마당에 목표주소를 넣겠는가, 목표명령을 넣겠는가 하는것은 잘 생각하여 합리적인 방안을 찾아야 한다. 목표주소를 기억하는것은 표의 크기를 작게 하지만 목표명령을 넣는 방식에 비해 더 큰 명령꺼내기시간을 요구한다[RECH98].

그림 11-17 은 이 현상을 절대로 일어 나지 않는다고 예측하는 방법인 경우에 대하여 보여 주고 있다. 목표주소를 넣는 방식에서는 명령꺼내기단계가 늘 다음의 련속되는 주소를 불러 낸다. 분기가 있으면 처리장치에서 일부 론리요소가 이것을 검출하여 다음명령을 목표주소로부터 불러 들일것을 지시한다. 분기경력표는 캐쉬로서 취급된다. 모든 미리꺼내기는 분기경력표에서 검사를 시작한다. 일치하는것이 발견되지 않으면 다음순서의 주소를 불러 들인다. 이 주소에 대한 일치가 있으면 예측은 명령의 상태에 기초하여 이루어 진다. 즉 다음에 련속되는 주소 혹은 분기목표주소가 선택론리요소에 공급된다.

분기명령이 실행되면 실행단계는 그 결과와 함께 분기경력표를 론리요소에 신호한다. 명령의 상태는 정확한 혹은 비정확한 예측을 되돌리도록 갱신된다. 예측이 비정확한 경우는 선택론리가 다음꺼내기를 위한 정확한 주소를 다시 가리켜 준다. 조건분기명령이 이 표내에 없다는것이 확인되면 그것이 표에 추가되며 출구기입들중의 하나를 제 4 장에서 고찰한 캐쉬재배치알고리듬들중의 어느 하나를 리용하여 버린다.

고급한 마이크로장치인 AMD29000 극소형처리장치에서는 분기경력표를 리용하고 있다.

지연분기

프로그람내에서 명령들을 자동적으로 재배렬하면 관흐름의 성능을 개선할수 있다. 이렇게 되면 분기명령들이 실제적으로 자기 시간보다 후에 지연되여 일어 날수 있다. 이와 같은 지연된 분기를 꾸미는 방법은 제 12 장에서 구체적으로 고찰한다.

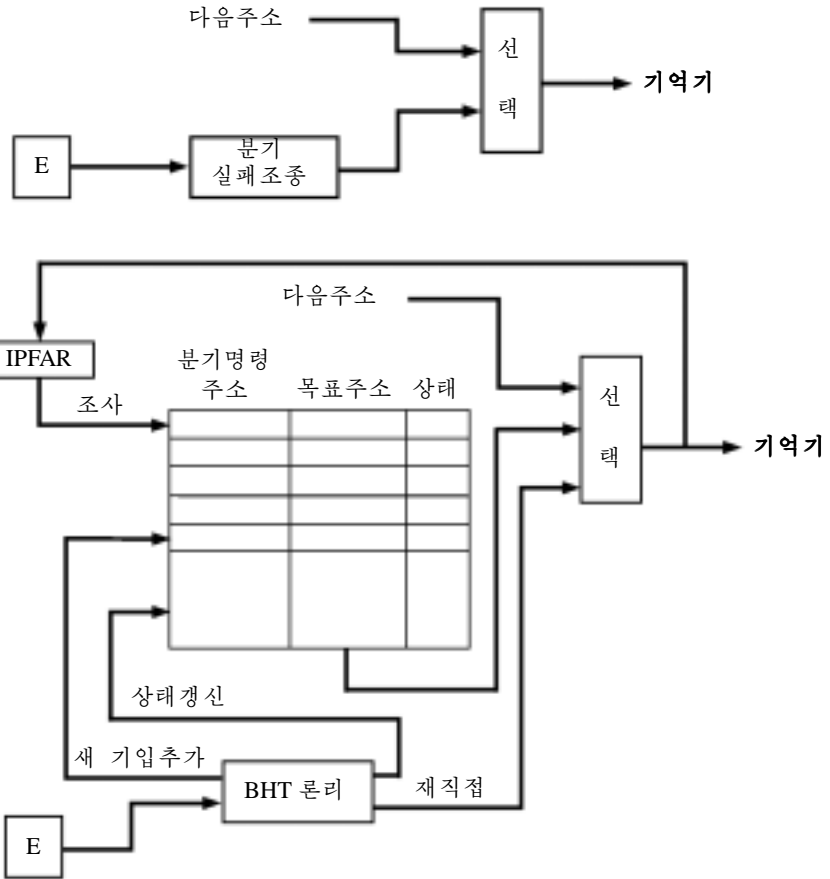



그림 11-17. 분기처리

4. INTEL 80486 의 관흐름처리

80486 은 5 단계로 된 관흐름을 가지고 있다.

- **꺼내기** : 이 단계에서는 명령들을 캐쉬 혹은 외부기억기에서 불러 내여 두개의 16byte 미리꺼내기완충기중 어느 하나에 넣는다. 이 단계의 목적은 낡은 자료가 명령해신기에 의해 소비되자마자 새 자료를 미리꺼내기완충기에 넣는것이다. 명령들이 앞붙이를 계산하지 않고도 1 부터 11byte 까지의 범위내에서 그 길이가 변하므로 다른 관흐름단계와 관련되는 미리꺼내기상태는 명령에 따라 변한다. 평균적으로 대략 5 개의 명령들을 16byte 씩 넣으면서 불러 낸다. 꺼내기단계는 미리꺼내기완충기를 꽉 채우도록 다른 단계와는 독립적으로 동작한다.
- **해신단계 1** : 이 단계에서는 모든 조작코드와 주소지정방식정보를 해신한다. 이 단계를 간단히 D1 단계라고도 한다. 명령길이정보와 같은 정보는 많아서 명령의 첫 3 byte 에 포함된다. 따라서 이 3byte 를 미리꺼내기완충기에서 꺼내여 해신한다. D1 해신기는 그다음 D1 해신에 참가하지 않은 명령의 나머지부분(변위와 직접값자료) 을 D2 단계에 보낸다.

- **해신단계 2**: 이 단계에서는 매 조작코드를 ALU에 필요한 조종신호로 확장한다. 또한 복잡한 주소지정방식들의 계산을 조종한다.
- **실행** : 이 단계는 ALU연산, 캐쉬호출,등록기갱신 등의 동작을 수행한다.
- **뒤로쓰기**: 이 단계는 필요한 경우 앞의 실행단계에서 변경된 등록기 및 상태기발을 갱신한다. 현 명령이 기억기를 갱신한다면 계산된 값을 캐쉬와 모선대면부쓰기완충기에 동시에 보낸다.

두 해신단계를 리용하면 관흐름이 박자주기당 하나의 명령에 가까운 처리능력을 유지할수 있다. 복잡한 명령들이나 조건분기명령들은 이 속도를 보장할수 없다.

그림 11-18은 관흐름의 동작실례를 보여 주고 있다. 이 그림에서 ㄱ는 기억기호출을 요구할 때 관흐름에서 생기는 지연이 없다는것을 보여 주고 있다. 그러나 이 그림의 ㄴ에서 보는바와 같이 기억기주소를 계산하는데 리용된 값인 경우에는 지연이 있을수 있다. 즉 값을 기억기로부터 등록기로 넣고 그다음 그 등록기를 다음명령의 기준등록기로 리용하는 경우에는 처리장치가 한주기동안 정지한다. 이 실례에서 처리장치는 첫번째 명령의 실행단계에서 캐쉬를 호출하고 뒤로쓰기단계기간에 등록기에 씌여 진 값을 기억한다. 그런데 다음명령이 해신단계 2에서 이 등록기를 요구한다. 해신단계 2가 앞선 명령의 뒤로쓰기단계와 나란히 서게 되면 우회신호경로는 해신단계 2 가 뒤로쓰기단계에서 리용하고 있는 같은 자료에 대한 호출을 진행하도록 하여 관흐름의 한 단계를 절약한다.

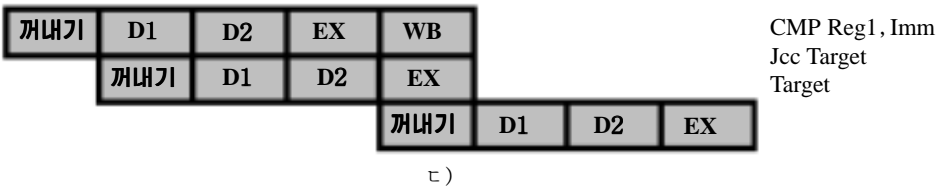
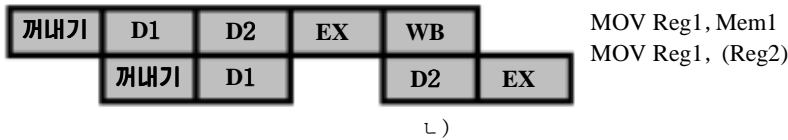
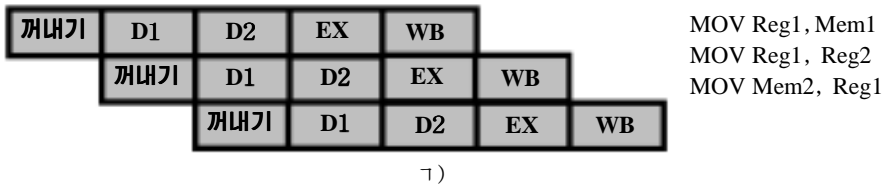



그림 11-18. 80486명령흐름선실례
ㄱ-흐름선에서 자료넣기지연이 없다, ㄴ-지시기넣기지연, ㄷ-분기명령동기

그림 11-18 ㄷ는 분기가 있다고 가정한 경우 분기명령의 시간일치(동기화)를 보여 주고 있다. 비교명령은 뒤로쓰기단계에서 조건코드를 갱신하여 뛰여넘기명령의 실행단계에서 우회경로가 이 조작코드를 리용할수 있도록 한다. 이와 병행하여 처리장치는 뛰여넘

기명령의 실행단계기간 뛰여넘기목표에서 추상적인 꺼내기주기를 실행한다. 처리장치가 거짓분기조건을 판정하면 이 미리꺼내기는 취소되고 다음에 련속되는 명령의 실행을 계속한다(이미 불러 내여 해신하였다.).

제 5 절. Pentium 처리장치

Pentium Ⅱ처리장치조직과 관련한 개괄은 이미 그림 4-23 을 통하여 보았다. 이 절에서는 일부 보다 상세한 점들을 고찰한다.

1. 등록기의 조직

등록기조직은 다음과 같은 형의 등록기들을 가지고 있다(표 11-1).

- **일반형** : 8 개의 32bit 일반등록기들이다(그림 11-3 ㄷ 참고). 등록기들은 모든 형의 Pentium Ⅱ명령들에 리용할수 있다. 또한 주소계산을 위한 연산수를 가질수 있다. 이외에 일부 등록기들을 특별한 목적에 전용으로 쓸수 있다. 실례로 문자렬명령은 명령에서 이 등록기들을 명시적으로 참조하지 않고도 연산수로서 리용할수 있는 ECX, ESI, EDI 등록기들의 내용을 리용한다. 결국 여러개의 명령들을 보다 함축하여 프로그람을 작성할수 있다.
- **토막**: 6 개의 16bit 토막등록기들은 토막선택기들을 가지는데 이 토막선택기들은 제 7 장에서 고찰한바와 같이 토막표들에서 첨수를 가리킨다. 코드토막(CS)등록기는 실행되고 있는 명령을 가지는 토막을 참조한다. 탄창토막(SS)등록기는 사용자가 볼수 있는 탄창을 가지는 토막을 참조한다. 나머지토막등록기들(DS, ES, FS, GS)은 사용자가 한번에 4 개까지의 개별적인 자료토막들을 참조할수 있게 하는데 쓴다.
- **기발** : EFLAGS 등록기는 조건코드와 여러가지 방식비트들을 가진다.
- **명령지시기** : 현재명령의 주소를 가진다.

표 11-1. Pentium Ⅱ 처리장치의 등록기들 ㄱ)옹근수단위

형	수	길이(비트)	목적
일반	8	32	일반사용자등록기
토막	6	16	토막선택기를 가진다
기발	1	32	상태 및 조종비트
명령지시기	1	32	명령지시기

ㄴ) 류점수단위

형	수	길이(비트)	목적
수값	8	80	류점수를 가진다
조종	1	16	조종비트들
상태	1	16	상태비트들
꼬리표단어	1	16	수값등록기의 내용 규정
명령지시기	1	48	례외에 의해 중단된 명령을 지적
자료지시기	1	48	례외에 의해 중단된 연산수를 지적

등록기조직에는 또한 특별히 류점수단위로 동작하는 등록기들도 있다.

- **수값 :** 매 등록기는 확장된 정확도를 가진 80bit 의 류점수를 가진다. 여기에는 밀어넣기와 밀어꺼내기와 같은 연산을 진행하며 탄창으로서의 기능을 수행하는 8 개의 등록기들이 있다.
- **조종 :** 16bit 의 조종등록기는 반올림조종의 형들인 단정확도, 배정확도, 확장된 정확도를 가진 류점수단위의 연산을 조종하는 비트들과 여러가지 례외조건들을 가능 혹은 불가능하게 하는 비트들을 가진다.
- **상태 :** 16bit 의 상태등록기는 탄창의 꼭대기에 있는 3 bit 의 지시기를 포함하여 류점수단위의 현재상태를 반영하는 비트들을 가진다. 3 bit 의 지시기는 마지막명령의 결과를 보고하는 조작코드와 례외기발들을 가진다.
- **꼬리표단어 :** 이 16bit 의 등록기는 모든 류점수값등록기에 대하여 두비트의 꼬리표를 가진다. 이 꼬리표는 대응하는 등록기내용의 속성을 지적한다. 속성에는 4 가지 가능한 값들 즉 유효, 령, 특별(NaN, 무한대, 비정규화), 빔이 있다. 이 꼬리표들은 프로그람들이 등록기내에서 실지자료의 복잡한 해신을 수행함이 없이 수값등록기의 내용을 검사할수 있게 한다. 실례로 전후관계에서 변화가 일어 나면 처리장치는 빈 임의의 류점수등록기를 보관할 필요가 없다.

이상에서 고찰한 등록기들의 리용은 쉽게 리해가 된다. 이제 여러개의 등록기들을 보다 상세히 고찰해 보자.

EFLAGS 등록기

EFLAGS 등록기(그림 11-19)는 처리장치의 조건을 가리키며 연산을 조종하도록 돕는다. 이 등록기에는 표 9-8 에서 정의한 6 개의 조건코드 즉 자리올림, 기우성, 보조자리올림, 령, 부호, 자리넘침조건코드들이 포함되여 있다. 이것은 옹근수연산결과에 대한 보고라고 볼수 있다. 이외에 조종비트들로서 다음과 같은것들이 있다.

- **내부새치기기발(TF):** 설정되면 매 명령의 실행후에 새치기가 발생한다. 오유수정에 리용한다.
- **새치기가능기발(IF) :** 설정되면 처리장치는 외부새치기를 접수한다.
- **방향기발(DF) :** 문자렬처리명령이 16bit 의 반등록기 SI 와 DI(16bit 연산인 경우) 혹은 32bit 의 등록기 ESI 와 EDI(32bit 연산인 경우)를 증가 혹은 감소시키는가를 결정한다.
- **I/O 특권기발(IOPL):** 설정되면 처리장치가 보호방식동작기간 I/O 장치에 대한 모든 호출에서 례외를 발생시키도록 한다.
- **재개기발(RF) :** 프로그람작성자가 명령이 오유수정후에 또 다른 오유수정을 즉시 발생시킴이 없이 다시 시작될수 있도록 오유수정을 불가능으로 만든다.
- **일치검사(AC):** 단어 혹은 배단어가 비단어 혹은 비배단어경계로 지정된 경우 설정된다.
- **식별기발(ID):** 이 비트가 설정되여 지워 지면 처리장치가 CPUID 명령을 지원하고 있다는것을 가리킨다. CPUID 명령은 회사, 계렬, 모형에 대한 정보를 준다.

이외에도 동작방식과 관련한 4 개의 비트들이 있다. 겹쌓인 과제(NT:Nested Task)기발은 현재과제가 보호방식동작으로 다른 과제내에 있다는것을 지시한다. 가상방식(VM:Virtual Mode)비트는 프로그람작성자가 가상 8086 방식을 가능 혹은 불가능하게 한다. 가상 8086 방식은 처리장치가 8086 기계로 동작하겠는가를 결정한다. 가상새치기기발

(VIF)과 가상새치기미정(VIP:Virtual Interupt Pending)기발은 다중과제처리환경에서 리용한다.

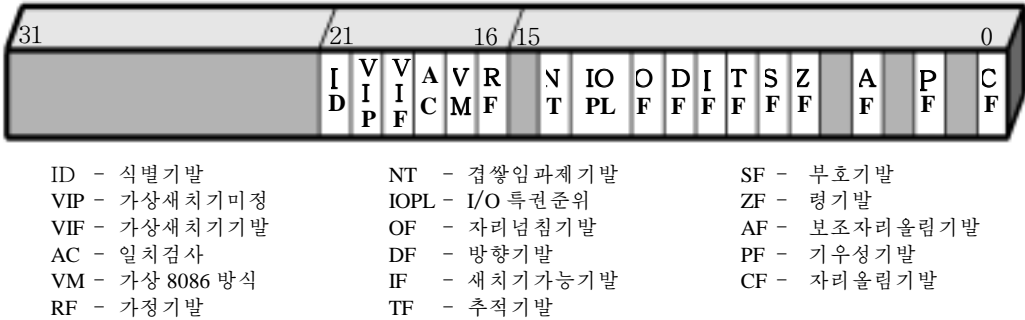



그림 11-19. Pentium Ⅱ EFLAGS 등록기

조종등록기

Pentium Ⅱ는 처리장치동작의 각이한 상황을 조종하는 4개의 32bit조종등록기(등록기 CR1 은 리용되지 않는다.)를 가지고 있다(그림 11-20). CR0 등록기는 체계조종기발로서 개별적인 과제의 실행을 위해서가 아니라 일반적으로 처리장치에 적용하는 방식을 조종하거나 상태를 지적한다. 기발들은 다음과 같다.

- **보호가능**(PE): 동작의 보호방식을 가능/불가능하게 한다.
- **감시협동처리장치**(MP:Monitor Coprocessor): Pentium Ⅱ의 이전 계렬의 콤퓨터들로부터 프로그람을 집행하고 있을 때에만 유용하다. 산수연산협동처리장치의 존재와 관련된다.
- **모방**(EM): 처리장치가 류점수단위를 가지지 않을 때 설정되며 류점수명령을 실행하려는 시도가 있을 때 새치기를 발생한다.
- **과제절환**(TS): 처리장치가 과제를 절환하였다는것을 가리킨다.
- **확장형**(ET): Pentium Ⅱ에서는 리용되지 않는다. Pentium Ⅱ이전 계렬의 기계들에서 수학협동처리장치명령의 지원을 가리키는데 리용된다.
- **수값오유**(NE): 외부모선들에서 류점수오유를 보고하는 표준기구를 가능하게 한다.
- **쓰기보호**(WP): 이 비트가 지워지면 읽을수만 있는 사용자준위페지들을 체계관리자처리에 의해 쓰기가 가능하게 할수 있다. 이 성질은 일부 조작체계들에서 처리창조를 지원하는데 쓸모가 있다.
- **일치마스크**(AM): 일치검사를 가능/불가능하게 한다.
- **쓰기불가능**(NW): 자료캐쉬의 동작방식을 선택한다. 이 비트가 설정되면 자료캐쉬를 쓰기동작으로부터 금지시킨다.
- **캐쉬불가능**(CD): 내부캐쉬채우기기구를 가능/불가능으로 한다.
- **페지화**(PG): 페지화를 가능/불가능으로 한다.

페지화가 가능하게 되면 CR2와 CR3 등록기는 유효이다. CR 등록기는 페지결함새치기가 발생하기전에 마지막으로 호출한 페지의 32bit선형주소를 가진다. CR3의 맨 웃자리비트로부터 20 개의 비트들은 페지등록부의 20 개의 맨 웃자리비트로 된 기준주소를 보유한다. 이 주소의 나머지부분은 0 으로 된다. CR3 의 두개의 비트는 외부캐쉬의 동작을 조종하는

단자들을 구동하는데 리용된다. 페지준위의 캐쉬불가능(PCD)비트는 외부캐쉬를 가능 혹은 불가능으로 만들며 페지준위쓰기투과(PWT)비트는 외부캐쉬를 통하여 쓰기를 조종한다.

그림 11-20. Pentium Ⅱ조종등록기

CR4 에는 다음과 같은 9 개의 추가적인 조종비트들이 있다.

- **가상 8086 방식확장(VME)** : 가상 8086 방식에서 가상새치기기발에 대한 지원을 가능하게 한다.
- **보호방식가상새치기(PVI)** : 보호방식에서 가상새치기기발에 대한 지원을 가능하게 한다.
- **시간표식(압인)불가능(TSD)** : 시간표식계수기로부터의 읽기(RDTSC)명령을 불가능하게 한다. 이 명령은 오유수정목적에 리용된다.
- **오유수정확장(DE)** : I/O 중지점을 가능으로 한다. 이것은 처리장치가 I/O 읽기와 쓰기를 중단하도록 한다.
- **페지크기확장(PSE)** : Pentium 에서 설정되면 4Mbyte 의 페지, Pentium Pro 와 Pentium Ⅱ에서 설정되면 2 Mbyte 의 페지리용을 가능하게 한다.
- **물리주소확장(PAE)** : PSE 에 의해 조종되는 특별한 새로운 주소지정방식이 Pentium Pro 와 Pentium Ⅱ에 대하여 가능으로 되였을 때는 주소선 A35 로부터 A32 까지를 언제나 가능으로 한다.
- **기계검사기능(MCE)**: 기계검사새치기를 가능하게 한다. 이 새치기는 자료기우성오유가 읽기모선주기기간에 일어 나거나 모선주기가 성공적으로 완료되지 않을 때 일어 난다.
- **페지전역가능(PGE)** : 총체적인 페지의 리용을 가능하게 한다. PGE=1 이고 과제절환이 수행되면 모든 TLB 기입은 표식이 붙은 전역을 제외하고 넘쳐 나게 된다.

- **성능계수기가능(PCE):** 임의의 특권준위에서 성능계수기읽기(RDPMC)명령의 실행을 가능하게 한다. 두개의 성능계수기들은 특별한 형의 사건의 존속시간과 사건의 발생수를 재는데 리용된다.

MMX 등록기

Pentium II MMX 가 여러가지 64bit 자료형을 만드는 능력을 가지고 있다는것은 이미 제 9 장 제 4 절에서 고찰하였다. MMX 명령들은 3bit 의 등록기주소마당을 리용하여 8 개의 MMX 등록기들을 식별한다.실지로 처리장치는 전용 MMX 등록기들을 포함하고 있지 않다. 오히려 처리장치는 뒤섞는 기술을 리용한다(그림 11-21). 존재하는 류점수등록기들은 MMX 값들을 기억하는데 리용된다.

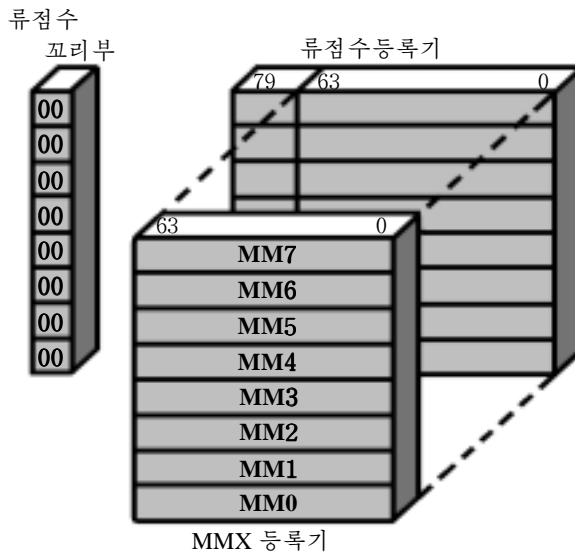



그림 11-21. MMX 등록기의 류점수등록기로의 배치

이 등록기들의 MMX 리용에서는 다음과 같은 기본특성이 있다.

- 류점수등록기들은 류점수연산을 위한 탄창으로서 취급된다. MMX 연산동작에서는 이와 같은 등록기들에 직접 호출할수 있다.
- MMX 명령은 임의의 류점수연산을 수행한 다음에 실행되며 FP 꼬리표단어가 유효하게 표식이 붙게 된다. 이것은 등록기주소지정을 가리키는 탄창동작으로부터의 변화를 반영한다.
- EMMS(빈 MMX 상태)명령은 모든 등록기들이 비였다는것을 지시하도록 FP 꼬리표단어의 비트들을 설정한다. 프로그람작성자는 련속되는 류점수연산이 알맞게 수행되도록 MMX 코드블로크의 끝에 이 명령을 삽입하는것이 좋다.
- 값을 MMX 등록기에 쓰면 대응하는 FP 등록기(부호와 지수비트들)의 비트들 [79:64]은 모두 1 로 설정된다. 이것은 류점수값으로 고찰될 때 NaN(수가 아니다.) 혹은 무한대로 FP 등록기에 값을 설정한다. 이것은 MMX 자료값이 유효한 류점수값처럼 보이지 않도록 한다.

2. 새치기처리

처리장치에서 새치기처리는 조작체계를 지원하도록 제공되는 기능이다. 새치기처리는 각이한 새치기조건에 맞게 응용프로그람집행을 잠시 중지한후에 다시 그 실행이 회복되게 한다.

새치기와 례외

새치기와 례외는 Pentium Ⅱ가 현재명령흐름의 실행을 중지하고 사건에 응답하도록 한다. 이 두 경우에 처리장치는 현재 처리의 전후상황을 보관하고 조건에 따라서 이미 정의된 루틴으로 처리를 옮긴다. **새치기**는 하드웨어로부터의 신호에 의해 발생하며 프로그람이 실행되는 동안 우연히 발생할수 있다. **례외**는 쏘프트웨어로부터 발생하며 명령실행과정에 일어 난다. 새치기와 례외에는 각각 두가지 발생원들이 있다.

① 새치기

- **금지형새치기**: 처리장치의 INTR 단자로부터 접수된다. 처리장치는 새치기가능기발(IF)이 설정되지 않는 한 금지형새치기를 인식하지 못한다.
- **비금지형새치기**: 처리장치의 NMI 단자로부터 접수된다. 이 형의 새치기는 무조건 접수된다.

② 례외

- **처리장치-검출례외**: 처리장치가 명령실행을 시도할 때 오유를 만나게 되면 그때 생긴다.
- **프로그람화된 례외**: 례외를 발생시키는 명령들에 의해 생긴다(INTO, INT3, INT와 BOUND).

새치기벡토르표

Pentium Ⅱ에서의 새치기처리는 새치기벡토르표를 리용한다. 모든 형의 새치기에는 번호가 할당되며 이 번호를 새치기벡토르표의 첨수로 리용한다. 이 표는 256개의 새치기벡토르들을 가지고 있다. 이것은 해당 새치기번호에 대한 새치기봉사루틴의 주소(토막과 변위)이다.

표 11-2는 새치기벡토르표에서 번호배치를 보여 주고 있다. 이 표에서 그늘진 곳에 기입된것은 새치기, 그늘이 지지 않는 곳에 기입된것은 례외를 의미한다. NMI(비금지형새치기)하드웨어새치기는 형이 2이다. INTR 하드웨어새치기들은 32로부터 255범위내의 번호에 할당된다. INTR 새치기가 발생하면 이 새치기에 대한 새치기벡토르번호가 모선우에 나타난다. 나머지벡토르번호들은 례외에 리용된다.

하나이상의 례외 혹은 새치기가 걸리는 경우 처리장치는 예측할수 있는 순서로 이것들에 대한 봉사를 한다. 표내에서 벡토르번호의 위치는 우선권을 반영하지 않는다. 례외와 새치기들의 우선권은 5개의 등급으로 나누어 진다. 우선권이 낮아 지는 순위로 이것을 고찰하면 다음과 같다.

- **등급 1**: 앞명령에 대한 내부새치기들(벡토르번호 1)
- **등급 2**: 외부새치기(2, 32～355)
- **등급 3**: 다음명령꺼내기에서의 고장(3, 14)
- **등급 4**: 다음명령해신에서의 고장(6, 7)
- **등급 5**: 명령실행에서의 고장(0,4,5,8,10～14,16,17)

표 11-2. Pentium II의 례외 및 새치기벡토르표

벡토르번호	설 명
0	나누기오유: 나누기자리넘침 혹은 령에 의한 나누기
1	오유수정례외: 오유수정과 관련한 여러가지 고장과 내부새치기를 포함한다.
2	NMI 단자새치기: NMI 단자에 신호한다.
3	중단점: INT3 명령에 의해 일어 난다. 이 명령은 오유수정에 효과적인 1byte 명령이다.
4	INT0-검출자리넘침: 처리장치가 OF 기발이 설정된 상태에서 INT0을 실행할 때 일어 난다.
5	BOUND 범위초과: BOUND 명령은 기억기에 기억된 경계와 등록기를 비교하여 등록기의 내용이 한계를 벗어 나는 경우 새치기를 발생한다.
6	조작코드를 정의하지 않는다.
7	리용할수 없는 장치: 외부장치의 부족으로 인하여 ESC 혹은 WAIT 명령을 리용하려는 시도가 실패한다.
8	2중 고장: 두개의 새치기가 같은 명령기간에 발생하여 련속적으로 조종될수 없다.
9	예약
10	무효과제상태포착: 요구되는 과제를 사용하는 토막이 초기화되지 않았거나 혹은 무효이다.
11	토막비존재: 필요한 토막이 존재하지 않는다.
12	탄창고장: 탄창토막의 경계를 벗어 났거나 탄창토막이 존재하지 않는다.
13	일반보호: 또 다른 례외가 일어 나지 않도록 하는 보호위반(즉 읽기만 하는 토막에 대한 쓰기)
14	페지결함
15	예약
16	류점수오유: 류점수산수연산명령에 의해 발생한다
17	일치검사: 기수바이트주소에 기억된 단어 혹은 4의 배수가 아닌 주소에 기억된 배단어에 대한 호출
18	기계검사: 모형규정
19~31	예약
2~255	사용자새치기벡토르: INTR 신호가 능동일 때 주어 지게 된다.
밝은 부분:	례외
그늘진 부분:	새치기

새치기조종

CALL 명령을 리용한 실행흐름에서와 같이 새치기조종루틴에로의 이행은 처리장치상태를 기억하는 체계탄창을 리용한다. 새치기가 발생하여 처리장치에 인식되면 다음의 사건들이 련속적으로 일어 난다.

- 이행이 특권준위의 변화를 동반한다면 탄창토막등록기와 확장된 탄창지시(ESP)등록기의 현재값을 탄창에 밀어 넣는다.
- EFLAGS 등록기의 현재값을 탄창에 밀어 넣는다.
- 새치기(IF) 및 내부새치기(TF) 기발을 모두 지운다. 이것은 INTR 새치기와 내부새

치기 혹은 단일계단특징을 불가능으로 하기 위한것이다.

- 현재코드토막(CS)지시기와 현재명령지시기(IP 혹은 EIP)를 탄창에 밀어 넣는다.
- 새치기가 오유코드에 의해 발생되면 오유코드를 탄창에 보관한다.
- 새치기벡토르내용을 불러 내여 CS와 IP 혹은 EIP 등록기들에 넣는다. 실행을 새치기봉사루틴으로부터 계속한다.

새치기로부터의 복귀를 위해 새치기봉사루틴은 IRET명령을 실행한다. 이 명령은 탄창에 보관된 모든 값들을 복귀시켜 새치기발생시점으로부터 실행을 재개한다.

제6절. PowerPC처리장치

PowerPC 처리장치조직에 대한 총적인 고찰은 그림 4-25에서 진행하였다. 이 장에서는 이 처리장치의 64bit실현에서 나서는 일부 구체적인 내용들을 고찰한다.

1. 등록기조직

그림 11-22에 PowerPC의 사용자가 볼수 있는 등록기들을 보여 주었다. 고정수등록기에는 다음과 같은것들이 있다.

- **일반**: 32개의 64bit일반등록기들이다. 이 등록기들은 자료연산수들을 넣기, 기억, 조작하는데 리용하며 또 등록기간접주소지정에도 리용할수 있다. 등록기0은 어느 정도 여러가지 목적에 리용할수 있다. 연산을 넣기 및 기억하거나 여러개의 더하기명령에서 그의 실지내용에는 관계없이 상수값 0을 가지는것으로 취급된다.
- **례외등록기(XER)**: 옹근수산수연산의 례외를 보고하는 3개의 비트들을 포함한다. 또한 일부 문자렬명령에서 연산수로서 리용되는 바이트계수마당도 가진다(그림 11-23 ㄱ).

류점수등록기로는 다음과 같은 사용자가 볼수 있는 등록기들을 리용한다.

- **일반**: 여기에는 32개의 64bit 일반등록기가 있으며 이것들은 모든 류점수연산에 리용된다.
- **류점수 상태 및 조종등록기(FPSCR)**: 이 32bit의 등록기는 류점수등록기의 연산을 조종하는 비트들과 류점수연산에서 생기는 상태를 보고하는 비트들을 가진다(표 11-3).

분기처리등록기로는 사용자가 볼수 있는 동록기들이 될수 있다.

- **조건등록기**: 8개의 4bit조건코드마당으로 구성된다(그림 11-23 ㄴ).
- **련결등록기**: 이 등록기는 목표주소의 간접주소지정을 위한 조건분기명령에 리용한다. 또한 호출 및 복귀동작에도 리용한다. 조건분기명령에서 LK 비트가 설정되면 그때 분기명령에 뒤따르는 주소는 련결등록기에 넣어 지며 후에 복귀용으로 쓸수 있다.
- **계수등록기**: 이 등록기는 제9장에서 설명한바와 같이 반복고리를 조종하는데 리용한다. 계수등록기는 조건분기명령에서 검사될 때마다 매번 하나씩 감소한다. 이 등록기는 분기명령에서 목표주소의 간접주소지정에도 리용된다.

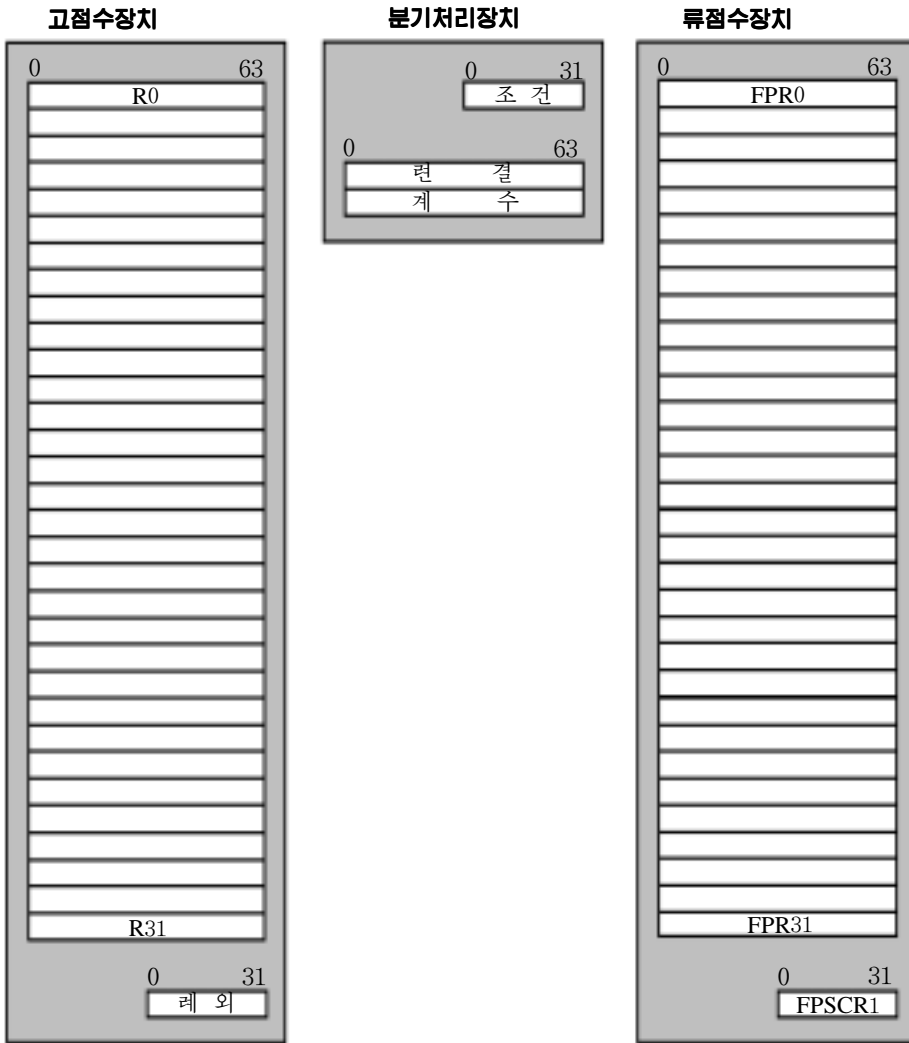



그림 11-22. PowerPC 의 사용자가 볼수 있는 등록기

조건등록기마당은 각이하게 리용할수 있다. 첫 4 개의 비트(CRO)는 Rc 비트가 설정되는 모든 옹근수산수연산명령들에 대하여 설정된다. 표 11-4 에서 보여 주는바와 같이 이 마당은 연산결과가 정수,부수 혹은 령인가를 가리킨다. 4 번째 비트는 XER 로부터 총적자리넘침비트를 복사한것이다. 그다음마당(CR1)은 Rc 비트가 설정되는 모든 류점수연산명령에 대하여 설정된다. 이 경우 4 개의 비트들은 FPSCR(표 11-3)의 첫 4 개의 비트들과 같게 설정된다. 결국 8 개의 조건마당(CR0~CR7)은 비교명령에 리용된다. 매 경우에 마당의 식별은 명령 그자체에서 규정된다. 고점수 및 류점수비교명령에서 지정된 조건마당의 첫 3 개의 비트들은 첫번째 연산수가 두번째 연산수보다 더 작은가, 더 큰가 혹은 두번째 연산수와 같은가 하는것을 나타낸다. 4 번째 비트는 고점수비교인 경우는 총적자리넘침이며 류점수비교인 경우는 무순서지적자이다.

표 11-3. PowerPC 의 류점수상태 및 조종등록기

비 트	정 의
0	총적레외. 임의의 례외가 일어 나면 설정된다. 쏘프트웨어에 의해 재설정될 때까지 그 설정을 유지한다.
1	총적가능레외. 임의의 가능레외가 일어 난 경우에 설정된다.
2	총적레외의 부당한 연산. 부당한 연산레외가 일어 났을 때 설정된다.
3	자리넘침레외. 결과크기가 표시범위를 초과하였을 때 설정된다.
4	아래자리넘침레외. 결과를 정규화하기에는 지내 작다.
5	령나누기레외. 나누이는수가 령이고 나누어지는수는 유한크기의 령이 아닌 수이다.
6	비정확한 례외. 반올림한 결과가 중간결과와 차이나거나 혹은 자리넘침이 자리넘침레외불가능에서 일어 난다.
7-12	적당치 못한 례외. 7: 신호화 NaN, 8: (∞−∞), 9: (∞÷∞), 10: (0÷0), 11: (∞×0), 12: NaN 이 동반되는 ㄱ비트
13	반올림한 결수부: 결수부를 증가시킨 결과의 반올림하기
14	비정확한 결수부: 반올림한 결과가 결수부를 변화시키거나 혹은 자리넘침이 자리넘침레외불가능에서 일어 난다
15 - 19	결과기발: 4bit 의 코드가 더 작다, 더 크다, 같다, 무순서, 정적 NaN, ±∞, ±정규화, ±비정규화, ± 0 을 규정한다.
20	예약
21-23	적당치 못한 연산레외. 21: 쏘프트웨어요구, 22: 부수의 두제곱뿌리, 23: ∞ 혹은 NaN 와 같은 큰 수에 대한 옹근수변환
24	적당치 못한 연산레외가능
25	자리넘침레외가능
26	아래자리넘침레외가능
27	령나누기레외가능
28	비정확한 례외가능
29	비 IEEE 방식
30 - 31	반올림조종, 2 비트코드는 가장 가까운 쪽으로, 0 인쪽으로, +∞방향으로의 반올림을 가리킨다.

밝은 부분: 상 태 비 트
그늘진 부분: 조 종 비 트

SO-총적자리넘침: 명령실행기간 발생한 자리넘침지시를 위하여 1 로 설정, 쏘프트웨어로 재설정될때까지 1 유지
OV-자리넘침: 명령실행기간 자리넘침이 발생하면 1 로 설정, 자리넘침이 없게 되면 다음 명령에서 0 으로 재설정
CA-자리올림: 명령실행기간 비트 0 밖으로의 자리올림이 발생하면 1 로 설정
바이트계수: 적재/기억문자렬첨수명령에서 바이트수를 지정

ㄱ)

ㄴ)

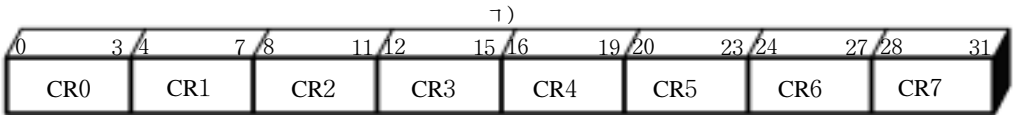



그림 11-23. PowerPC 의 등록기형식: ㄱ-고점수례외등록기, ㄴ-조건등록기

표 11-4. 조건등록기에서 비트들의 설명

비트위치	CR0(R_C=1 인 옹근수명령)	CR1(R_C=1 인 류점수명령)	CRi(고점수 비교명령)	CRi(류점수 비교명령)
i	결과<0	총적례외	연산수1<연산수2	연산수1<연산수2
I + 1	결과>0	총적례외가능	연산수1>연산수2	연산수1>연산수2
I + 2	결과=0	적당치 못한 연산의 총적례외	연산수1=연산수2	연산수1=연산수2
I + 3	총적자리넘침	자리넘침례외	총적자리넘침	무순서(한 연산수가 NaN 이다)

2. 새치기처리

임의의 처리장치에서와 마찬가지로 PowerPC 는 례외조건에 따라서 현재실행중에 있는 프로그람을 중단할수 있는 능력을 가진다.

새치기의 형

PowerPC 새치기에는 일부 체계조건이나 사건에 의하여 일어 나는것과 명령실행에 의하여 일어 나는것이 있다. 표 11-5 에 PowerPC 가 접수하는 새치기를 목록화하여 보여주었다.

표에서 보여 준 대다수의 새치기들은 쉽게 리해할수 있다. 체계재설정새치기는 전원을 투입하거나 재설정단추를 눌렀을 때 발생하는 새치기이다. 이 새치기는 체계를 재시동한다. 기계검사새치기는 캐쉬기우성오유나 존재하지 않는 기억기위치에 대한 참조 등 일부 비정상적인 현상과 관련하여 발생하는 새치기이다. 이 새치기는 체계를 검사정지상태 즉 처리장치의 실행을 중지하고 재시동이 있을 때까지 등록기의 내용들을 동결시키는 상태에 들어 가게 한다. 류점수방조새치기는 처리장치가 류점수장치에 의하여 직접 조종할수 없는 연산을 완료하도록 쏘프트웨어루틴을 조종할수 있게 한다. 류점수장치로 직접 조종할수 없는 연산에는 비정규화된 수들이나 실현할수 없는 류점수조작코드들에 의한것들이 속한다.

기계상태등록기(MSR)

프로그람의 중단에서 기본은 새치기가 발생하였을 때 처리장치의 상태를 회복시키는 능력이다. 여기에는 각이한 등록기의 내용뿐아니라 실행과 관련한 각이한 조종조건들도 포함된다. 이 조건들을 쉽게 MSR 에 모두 포함시킬수 있다(표 11-6).

이 등록기에 있는 여러개의 비트들을 간단히 설명하자. 특권방식비트(비트 49)가 설정되면 처리장치는 사용자특권준위에서 동작한다. 이때에는 명령모임의 한 부분묶음만을 리용할수 있다. 이 비트가 지워 지면 처리장치는 체계관리자특권준위에서 동작한다. 이것은 모든 명령들을 쓸수 있게 하며 사용자특권준위에서 호출할수 없었던 MSR 와 같은 일부 체계등록기들에 대한 호출도 할수 있게 한다.

단일계단추적비트(비트 53)가 설정되면 처리장치는 매 명령이 성과적으로 종결된후에 추적새치기조종기로 조종을 넘긴다. 분기추적비트(비트 54)가 설정되면 처리장치는 분기가 일어 났는가 일어 나지 않았는가에는 관계없이 매 분기명령이 성과적으로 종결된

후에 분기추적새치기조종기로 갈라 진다. 명령주소번역(비트 58) 및 자료주소번역(비트 59) 비트들은 실지주소지정이 리용되는가 혹은 기억기관리장치가 주소번역을 수행하는가를 결정한다.

표 11-5. PowerPC 의 새치기표

입구점	새치기형	설명
00000H	예 약	
00100H	체계재설정	처리장치의 하드 혹은 쏘프트를 재설정할것을 외부론리에 의해 신호한다.
00200H	기계검사	기계검사를 인식할수 있게 되면 처리장치에 TEA #을 확인
00300H	자료기억기	실레:자료페지결함:호출이 적제/기억에서의 위반을 바로 잡는다.
00400H	명령기억기	코드페지결함: I/O 토막으로부터의 명령꺼내기를 시도하였다. 호출이 위반을 바로 잡는다.
00500H	외부	외부새치기를 인식할수 있게 된 경우 외부론리에 의해 처리장치의 외부새치기를 입구하는것을 신호한다.
00600H	일치	부정합(일치하지 않음) 연산수로 인한 성공하지 못할 호출 시도
00700H	프로그람	류점수 새치기:특권명령을 실행하려는 사용자의 시도,지정된 조건을 만족시킬 때 실행되는 내부새치기명령,비법적인 명령
00800H	리용할수 없는 류점수	류점수장치불가능에서 류점수명령을 실행하려는 시도
00900H	감소기	외부새치기인식이 가능할 때 감소등록기의 고갈
00A00H	예약	
00B00H	예약	
00C00H	체계호출	체계호출명령의 실행
00D00H	추적	단일계단 혹은 분기추적새치기
00E00H	류동소수점수방조	상대적으로 드물고 복잡한 류점수연산(즉 비정규화된 수에 대한 연산) 을 하려고 하는 시도
00E10H 로부터 00FFFFH 까지	예약	
01000H 로부터 02FFFH 까지	예약 (특수한 실현)	

밝은 부분: 새치기들이 명령실행에 의해 일어 남
그늘진 부분: 새치기들이 명령실행에 의해 일어 나지 않음

류점수 례외 0	류점수 례외 1	인식되는 새치기들
0	0	인식되지 않음
0	1	비정확하며 회복할수 없음
1	0	비정확하며 회복할수 있음
1	1	정확함

표 11-6. PowerPC 의 기계상태등록기

비트	정의
0	처리장치가 32bit/64bit 방식에 있다
1:44	예약
45	전원관리 가능/불가능
46	실현관계(1520)
47	새치기조종기가 큰끝 혹은 작은끝배치방식에서 실행하고 있는가를 정:
48	외부새치기가능/불가능
49	특권/비특권상태
50	류점수장치 리용가능/리용불가능
51	기계검사새치기가능/불가능
52	류점수례외방식
53	단일계단추적가능 /불가능
54	분기추적가능/불가능
55	류점수례외방식 1
56	예약
57	례외주소의 맨 웃자리부분이 000h/FFFh
58	명령주소번역설정/차단
59	자료주소번역설정/차단
60:61	예약
62	새치기회복가능/회속불가능
63	처리장치가 큰끝/작은끝배치방식에 있다.

비그늘: SSR1 에 복사됨
그늘: SSR1 에 복사되지 않음

새치기조종

새치기가 발생하여 처리장치가 그것을 인식하면 다음과 같은 순서로 사건들이 일어 난다.

① 처리장치는 보관/회복등록기 0(SRR0)에 다음에 실행되는 명령의 주소를 넣는다. 이것은 명령실행에 대한 잘못된 시도로 새치기가 발생한 경우에 현재실행중에 있는 명령의 주소로 되며 그렇지 않은 경우는 현재명령다음에 실행되는 명령의 주소로 된다.

② 처리장치는 MSR 로부터 보관/회복등록기 1(SRR1)에 기계상태정보를 복사한다. 표 11-6 에서 그늘이 없는 부분에 씌여 진 비트들이 복사된다. SRR1 의 나머지비트들은 새치기형에 고유한 정보를 가지게 된다.

③ MSR 는 새치기형에 고유한 하드웨어-정의값으로 설정된다. 모든 새치기형에 대하여 주소번역이 금지되고 외부새치기가 불가능하게 된다.

④ 그다음 처리장치는 새치기조종기에로 조종을 넘긴다. 새치기조종기의 주소는 새치기표에 기억된다(표 11-5). 이 표의 기준주소는 MSR 의 비트 57 에 의해 결정된다.

새치기로부터 돌아 올 때에는 새치기봉사루틴이 rfi(새치기로부터의 되돌이) 명령을 실행한다. 이 명령은 SRR1 에 보관된 비트값들이 MSR 에 복귀되도록 한다. 실행은 SRR0 에 기억된 위치에서 재개된다.

참고문헌

[HENN91]과 [HWAN93]은 관흐름처리에 대하여 상세히 고찰하고 있다. [SOHI90]은 명령관흐름과 관련한 하드웨어설계문제점들을 매우 구체적으로 고찰하고 있다. [CRAG92]는 명령관흐름에서 분기예측에 대한 상세한 연구결과들을 서술하고 있다. [DUBE91]과 [LILJ88]은 명령관흐름처리의 성능을 강화하는데 리용할수 있는 여러가지 분기예측전략들을 고찰하고 있다. [KAEL91]은 목표주소가 변하는 명령들에 의한 분기예측의 곤난성을 고찰하고 있으며 [TABA91]은 Intel 80486 명령관흐름에 대하여 서술하고 있다.

[BREY97]은 Pentium 에 대하여, [SHAN95a]는 PowerPC 에 대하여 각각 새치기처리와 관련한 총체적인 고찰을 진행하고 있다.

BREY97 Brey, B. *The Intel Microprocessors: 8086/8066, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Processor.* Upper Saddle River, NJ:Prentice Hall, 1997.

CRAG92 Cragon, H. Branch Strategy Taxonomy and Performance Models. Los Alamitos, CA:IEEE Computer Society Press, 1992.

DUBE21 Dubey,P. and Flynn,M. "*Branch Strategies: Modeling and Optimization.*" *IEEE Transactions on Computers,* October 1991.

HENN91 Hennessy,J., and Jouppi,N. "Computer Technology and Architecture: An Evolving Interaction." *Computer*, September 1991.

HWAN93 Hwang,K. *Advanced Computer Architecture.* New York: McGraw-Hill, 1993.

KAEL91 Kaeli,d., and Emma, P.,"Branch History Table Prediction of Moving Target Branches Due to Subroutine Returns." *Proceeding, 18th Annual International Symposium on Computer Architecture,* May 1991.

LILJ88 Lilja, D. "Reducing the Branch Penalty in Pipelined Processors." *Computer*, July 1988.

SHAN95a Shanley, T. *PowerPC System Architecture.* Reading, MA:Addison-Wesley, 1995.

SOHI90 Sohi, G. "Instruction Issue Logic for High-Performance Interruptable, Multiple Functional Unit,Pipelined Computers." *IEEE Transactions on Computers,* March 1990.

TABA91 Tabak, D. *Advanced Microprocessors*. New York: McGraw-Hill, 1991.

련습문제

1. ㄱ. 콤퓨터에서 8bit 단어로 수행되는 마지막연산이 두 연산수가 각각 2 와 3 인 더하기였다면 다음기발의 값은 얼마인가?
   - 자리올림
   - 령
   - 자리넘침

- 부호
- 우수기우성
- 반자리올림

ㄴ. 연산수가 －1(2의 보수)과 ＋1이라면 이 기발들의 값은 얼마인가?

2. 그림 11-11의 시간선도를 고찰해 보자. 두 단계의 관흐름(꺼내기,실행)이 있다고 가정한다. 이제 4개의 명령에 대하여 몇시간단위가 요구되는가를 보여 주는 선도를 다시 그리시오.

3. 명령관흐름을 통하여 흐르고 있는 길이가 n인 명령렬을 생각하자. p는 조건 혹은 무조건분기명령을 만나는 확률, q는 분기명령 I의 실행이 련속이 아닌 주소로 이행하는 확률이라고 하자. 그리고 이와 같은 매 이행은 I가 마지막단계에서 나타 날 때 진행중인 모든 명령처리를 파괴하면서 제자리로 되돌아 가는 관흐름을 요구한다고 가정하자. p, q 확률들을 고려하여 식 11-1과 식 11-2를 다시 표시하시오.

4. 관흐름에서 분기들을 처리하는 여러흐름방법의 한가지 제한성은 첫 분기가 완전히 해결되기도전에 또 다른 추가적인 분기를 만나게 되는것이다. 두개의 추가적인 제한성 혹은 부족점을 제기하시오.

5. 그림 11-24의 상태도를 고찰해 보자.

ㄱ.상태도의 매 부분의 작용을 서술하시오.

ㄴ.이것을 제4절에서의 분기예측상태도와 비교하시오.

분기예측에 대한 세가지 방법들의 상대적인 우점들을 각각 론의해 보시오.

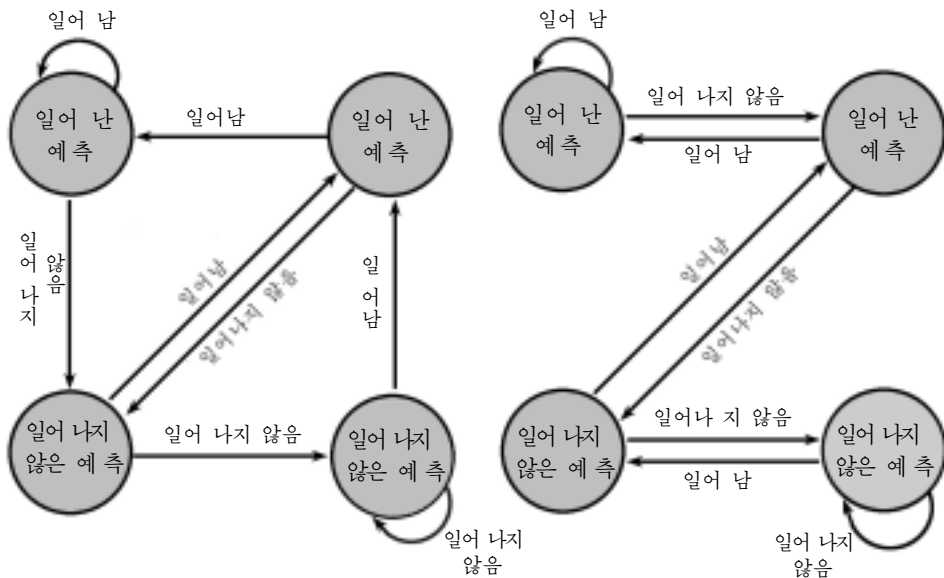



그림 11-24. 문제 6을 위한 상태도

6. Motorola 680x0 기계는 다음과 같은 양식의 조건에 따르는 감소 및 분기명령을 가지고 있다.

DBcc Dn, displacement

여기서 cc 는 검사할수 있는 조건들중의 하나이다. Dn 은 일반등록기이며 displacement 는 현재명령과 관련된 목표주소를 규정한다. 명령은 다음과 같이 정의될수 있다.

```
if (cc = False)
then   begin
       Dn:= (Dn) - 1;
       if Dn ≠-1 then PC := (PC) + displacement   end
else PC := (PC) + 2;
```

명령이 실행되면 우선 고리의 종결조건이 만족되는가를 판정하기 위하여 검사된다. 조건이 만족되면 연산이 수행되지 않고 다음명령에 대한 실행이 차례로 계속된다. 조건이 만족되지 않으면 지정된 자료등록기가 감소되여 그것이 령보다 더 작은가를 검사한다. 령보다 더 작다면 고리가 종결되고 차례로 다음명령에 대한 실행이 계속된다. 다른 한편 프로그람은 특별한 위치로 갈라 진다. 이제 다음과 같은 아쎔블리언어프로그람의 몇개의 렬을 생각하자.

```
AGAIN  CMPM.L    (A0)1, (A1)1
       DBNE   D1, AGAIN
       NOP
```

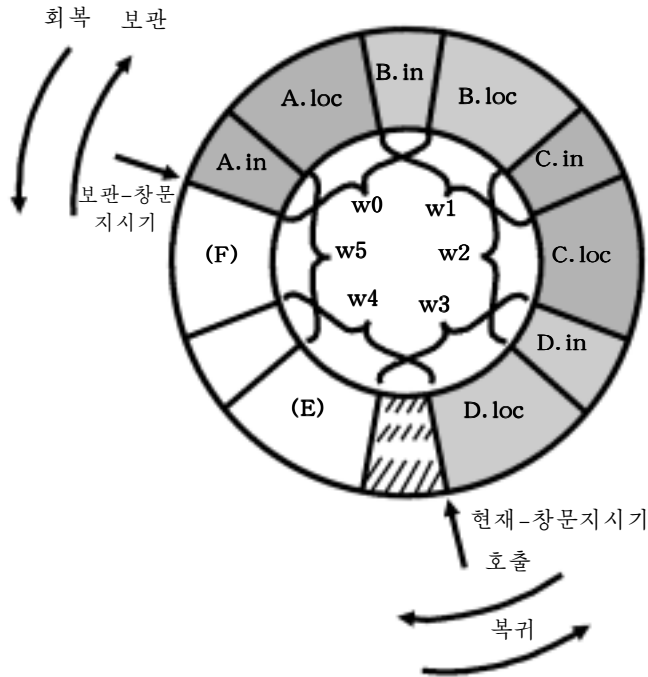
이 명령렬에서는 A0 과 A1 이라는 주소를 가진 두개의 문자렬이 같은가를 보기 위해 비교한다. 문자렬지시기는 참조를 할 때마다 증가된다. D1 은 처음에 비교되는 긴 단어(4 byte)의 수를 가진다.

ㄱ. 등록기들의 초기값이 A0=$00004000, A1=$00005000, D1=$000000FF ($는 16 진표기법을 의미한다.)이다. $4000 과 $6000 사이 기억기에는 단어 $AAAA 를 넣는다. 우의 프로그람이 실행중에 있는 경우 DBNE 고리가 실행되는 회수와 NOP 명령을 만났을 때의 세 등록기들의 내용을 구하시오.

ㄴ. $4000 와 $4FEE 사이의 기억기에는 $0000 를, $5000 과 $6000 사이의 기억기에는 $AAA 를 넣은 경우 ㄱ를 반복하시오.

7. 조건분기가 일어 나지 않는다고 가정하고 그림 11-18 ㄷ를 다시 그리시오.

제 12 장. 축소명령모임콤퓨터

◆◆◆

◆ 고급언어프로그람의 집행동작에 대한 연구는 새로운 형의 처리장치의 구성방식인 축소명령모임콤퓨터(RISC)를 설계하는데서 지름길을 마련해 주었다. RISC 에서는 값주기명령문이 기본을 이루고 있으며 단순한 자료의 전송도 최적화해야 한다고 주장하고 있다. 또한 많은 IF 나 LOOP 명령문들도 있는데 이것들도 효과적인 관흐름처리를 수행할수 있도록 기본순서조종기구를 최적화해야 한다고 보고 있다. 연산수참조현상에 대한 연구는 등록기에 적당한 수의 연산수를 대응시켜 유지하면 성능을 높일수 있다는것을 보여 주고 있다.

◆ 다음과 같은 RISC 기계의 기본특성은 이와 같은 연구결과로부터 얻어 진것이다.

① 고정된 형식으로 된 제한된 명령모임
② 많은 등록기 혹은 등록기사용을 최적화하는 콤파일러의 리용
③ 명령관흐름최적화의 강화

◆ RISC 의 단순한 명령모임은 명령당 수행되는 얼마간의 연산들을 예측할수 있으므로 효과적인 관흐름처리에 적합하다. 또한 RISC 명령모임구성방식은 지연분기처리방법을 리용하는데 적합하므로 관흐름의 효과성을 개선하는 다른 명령들과 함께 분기명령을 재배치할수 있는 우점이 있다.

◆◆◆

1950 년경에 기억프로그람콤퓨터가 개발된 때로부터 콤퓨터구성과 구성방식의 분야에서는 매우 놀랄만한 혁신들이 다소나마 이룩되였다. 아래에 콤퓨터의 산생으로부터 시작하여 일부 기본적인 진보에 대하여 개괄한다.

- **계렬화개념 :** 이것은 1964 년 IBM 이 System/360 에 도입한것이다. 그로부터 얼마후에 DEC 가 PDP-8 에 이 개념을 도입하였다. 계렬화개념은 콤퓨터의 구성방식을 그 실현과 분리시키는것이다. 여러가지 형태의 콤퓨터들은 사용자에게는 같은 구성방식을 주지만 각이한 단가, 각이한 성능을 가진다. 이와 같은 단가와 성능에서의 차이는 같은 구성방식에 기초하여 콤퓨터를 각이하게 만들기 때문이다.
- **마이크로프로그람화된 조종장치 :** 1951 년에 윌크스(Wilkes)에 의해 제안된것으로서 1964 년에 IBM 이 S/360 에 도입하였다. 마이크로프로그람작성은 조종장치의 설계 및 실현을 쉽게 하며 계렬화개념을 실현할수 있게 해준다.
- **캐쉬 :** 1968 년에 IBM S/360 Model 85 에 처음으로 도입되였다. 기억기계층에 이것을 삽입한 결과 놀라운 성능개선을 기대할수 있었다.
- **관흐름처리 :** 기계명령프로그람의 본질적인 순차속성(련속성)에 병렬처리개념을 도입한다는 의미를 가진다. 이 실례는 명령관흐름처리와 벡토르처리 등이다.
- **다중처리장치 :** 이것은 여러가지 각이한 구성들과 대상들을 망라한다.

이상에서 고찰한 콤퓨터구성방식에는 가장 흥미가 있으면서도 가장 중요한 혁신이라고도 볼수 있는 축소명령모임콤퓨터(RISC)구성방식이 포함되여 있어야 한다. RISC 구성방식은 처리장치구성방식의 력사에서 매우 놀랄만한것이다. RISC 구성방식에 대한 해석은 일반콤퓨터구성과 구성방식에서 제기되는 많은 중요한 문제점들에 중점을 두고 있다.

RISC처리장치를 여러 회사들에서 각이한 방법으로 그 정의를 하고 설계하였다고 해도 대부분 설계들에 공유된 기본요소들은 변함이 없다. 그 기본요소들은 다음과 같다.

- 큰 수의 일반등록기 혹은 등록기사용을 최적화하기 위한 콤파일러기술의 리용
- 제한된 단순한 명령모임
- 명령관흐름최적화의 강화

표 12-1 에서는 여러가지 RISC 체계와 비 RISC 체계들을 비교하고 있다.

이 장에서는 먼저 명령모임에 대한 일부 총괄적인 결과들을 간단히 보고 그다음 우에서 언급한 세가지 기본요소에 대하여 고찰한다.

표 12-1. 일부 CISC, RISC 와 슈퍼스칼라처리장치들의 특성

	복합명령모임콤퓨터(CISC)			**축소명령모임 콤퓨터(RISC)**		**슈퍼스칼라콤퓨터**		
특 성	IBM 370/168	VAX 11/780	Intel 80486	SPARC	MIPS R4000	PowerPC	Ultra SPARC	MIPS R10000
개발년도	1973	1978	1989	1987	1991	1993	1996	1996
명령크기(byte)	208	303	s	69	94	225		
주소지정	2-6	2-57	1-11	4	4	4	4	4
일반록기의 수	16	16	8	40-520	32	32	40-520	32
조종기억기크기(Kbit)	420	480	246	-	-	-	-	-
캐쉬크기(Kbit)	64	64	8	32	128	16-32	32	64

제1절. 명령실행특성

콤퓨터의 발전에서 제일 눈에 뜨이는것들중의 하나는 프로그람작성언어의 발전이다. 하드웨어의 비용은 떨어 지고 상대적으로 쏘프트웨어의 비용은 올라 갔다. 이와 함께 프로그람작성자들의 만성화된 편향은 절대적으로 쏘프트웨어값을 올리는것이다. 이로부터 체계의 생명주기에서 기본비용은 하드웨어가 아니라 쏘프트웨어로 되고 있다. 믿음성이 보장되지 못하고 있는 요소에는 비용뿐아니라 불편성도 있다. 다시말하여 체계 및 응용프로그람들인 경우 그것을 리용하는 시간이 지나감에 따라 새로운 결함들이 계속 나타나는것은 일반적인것이다.

이로부터 프로그람연구자들이나 프로그람산업계는 보다 강력하고 복잡한 고급프로그람작성언어를 개발하였다. 이 고급언어(HLL)들은 프로그람작성자가 알고리듬의 세부에 많은 주의를 돌리면서 보다 정확히 표시할수 있게 한다. 또한 본성적으로 구조화된 프로그람작성이나 객체지향설계를 리용할수 있게 한다.

이와 같은 해결방법에서는 **의미론적인 간격**이라고 하는 또 하나의 문제가 생긴다. 의미론적인 간격은 HLL에서 제공하는 연산들과 콤퓨터구성방식에서 제공하는 연산들사이의 차이를 말한다. 이 의미론적인 간격은 실행의 비효과성, 지나친 기계어프로그람크기, 콤파일러의 복잡성 등을 초래한다. 설계가들은 이 간격을 좁히는 구성방식을 개발하는것으로서 이에 대답하였다. 이렇게 만들어 진 구성방식들의 기본특징은 큰 명령모임과 여러가지 주소지정방식, 하드웨어로 실현된 각이한 HLL 명령문 등을 포함하고 있는것이다. 후자의 실례로서는 VAX에서의 CASE기계명령을 들수 있다. 이러한 복잡한 명령모임들은 다음과 같은것을 실현하는것을 목적으로 하고 있다.

- 콤파일러작성도구를 만드는 과제의 단축
- 복잡한 연산순서를 마이크로코드로 실현하여 실행효과성을 개선
- 한층 더 복잡하고 개선된 HLL 의 지원

한편 HLL 프로그람에서 산생되는 기계명령실행특성과 패턴을 정하기 위하여 많은 연구들이 수십년간 계속 되였다. 이 연구결과들은 보다 복잡한 HLL 이 아니라 보다 단순한 HLL을 지원하는 구성방식을 만들도록 하였다.

RISC 옹호자들의 론거를 리해하기 위하여 명령실행특성을 간단히 상기해 보면 흥미있는 평가측면은 다음과 같다.

- **수행된 연산들**: 이것들은 처리장치 및 처리장치와 기억기의 호상작용에 의해 수행되는 기능들을 결정한다.
- **리용된 연산수들**: 연산수형과 그 리용빈도는 연산수들을 기억하는 기억기구성과 연산수들에 호출하는 주소지정방식을 결정한다.
- **실행순서짜기**: 이것은 조종과 관흐름구성을 결정한다.

이 절의 나머지부분에서는 고급언어프로그람들에 대하여 진행한 여러가지 연구결과들을 개괄한다. 이 모든 결과들은 동적인 측정에 기초하고 있다. 즉 고급언어프로그람을 실행하여 일부 특징이 나타나거나 특별한 속성이 얻어 진 회수를 계수하는 방법으로 결과를 수집한다. 이와는 반대로 정적인 측정에서는 순수 프로그람의 원천본문에 대하여 이와 같은 계수를 진행한다. 따라서 이 방법은 매 명령문이 실행되는 회수와 관련한 의의 있는 정보를 얻지 못하므로 결국 성능에 대한 가치 있는 정보를 주지 못하게 된다.

1. 연산

HLL 프로그람들의 동작을 분석하기 위하여 여러가지 연구들이 진행되였다. 제 4 장에서 고찰한 표 4-9 는 이 연구들에서 기본결과들을 보여 주고 있다. 언어와 그 응용을 함께 고찰한 이 연구결과들에는 좋은 조언으로 될 결과들이 적지 않다. 우선 값주기명령문이 많이 리용되고 있으며 이것은 단순한 자료의 이동이 매우 중요하다는것을 말해 준다. 또한 조건명령문(IF, LOOP)들도 많이 리용되고 있는데 이 명령문들은 일정한 부류의 비교 및 분기명령을 가진 기계어로 실현되게 된다. 이것은 명령모임의 순서조종기구가 중요하다는것을 말해 준다.

이 결과들은 어떤 형의 명령문이 가장 많이 리용되며 이로부터 그 명령문에 대해서는《최적》양식을 반드시 구비해야 한다는것을 암시해 주므로 기계명령모임설계가들에게 중요한것으로 된다. 그러나 이 결과들은 일반적인 프로그람의 집행에서 어떤 명령문들이 가장 많이 리용되는가를 보여 주지 못하였다. 다시말하여 원천프로그람이 기계어프로그람으로 번역된 경우 원천프로그람에서 어느 명령문들이 기계어명령실행의 대부분을 차지하는가를 밝혀 내지 못하였다.

이와 같은 근본적인 현상을 밝혀 내기 위하여 부록 4-1 에서 서술한 Patterson 의 프로그람들[PATT82a]을 VAX, PDP-11 과 Motorola 68000 에서 콤파일하여 명령문종류마다 기계명령들의 평균수와 기억기참조를 평가하였다[표 12-2]. 표 12-2 에서 2 렬과 3 렬은 여러가지 프로그람들에서 각이한 HLL 명령들의 상대적인 출현빈도를 보여 주고 있다. 이 값들은 원천코드에서 명령문들이 발생하는 회수를 계수하는 방법으로가 아니라 프로그람을 실행시켜 출현회수를 계수하는 방법으로 얻은것들이다. 그러므로 동적인 빈도들의 통계라고 할수 있다. 4 렬과 5 렬의 값들 즉 기계명령들에 대한 자료는 2 렬과 3 렬에 있는 매 값에 콤파일러에 의해 얻어 진 기계명령들의 수를 곱하고 그다음 그것들을 정규화하여 상대적인 출현빈도로 나타낸것이다. 정규화는 HLL 명령문당 기계명령들의 개수를 고려하여 진행한다. 이와 류사하게 6 렬과 7 렬의 값들도 매 명령문당 발생하는 상대적인 기억기참조개수에 매 명령문형의 출현빈도를 곱하여 얻어 진다. 이상에서 본 4 렬로부터 7 렬까지의 자료들은 여러가지 형의 명령을 실행하는데 걸리는 실지시간을 대신 측정하여 얻은것들이다. 이 결과들은 HLL 프로그람들에서 일반적으로 틀호출 및 복귀가 가장 많은 시간을 소비하는 연산이라는것을 보여 주고 있다.

표 12-2 는 전형적인 현재명령모임구성방식에서 HLL 프로그람들을 번역하는 경우에 HLL 프로그람에서 여러가지 명령문형들의 중요성정도를 상대적으로 보여 주고 있다. 이와는 다른 구성방식인 경우에는 어느 정도 차이나는 결과들이 얻어 질수 있다. 그러나 이 연구는 오늘날의 복합명령모임콤퓨터구성방식들을 대표할수 있는 결과들을 주고 있다.

표 12-2. HLL 연산의 상대적인 동적빈도[PATT82 a]

	동적출현		기계명령		기억기참조	
	Pascal	C	Pascal	C	Pascal	C
ASSIGN	45	38	13	13	14	15
LOOP	5	3	42	32	33	26
CALL	15	12	31	33	44	45
IF	29	43	11	21	7	13
GOTO	-	3	-	-	-	-
OTHER	6	1	3	1	2	1

따라서 이 연구결과는 HLL 프로그람들에 대한 보다 효과적인 방도들을 찾는 사람들에게 어느 정도 지름길을 열어 주는것으로 될수 있다.

2. 연산수

연산수형의 출현에 대한 연구는 그것이 중요함에도 불구하고 훨씬 적게 진행되였다. 그렇지만 의의가 있는 여러가지 결과들이 있다.

이미 참고한 Patterson 의 연구[PATT82a] 역시 동적인 변수들의 등급발생빈도를 주의깊이 고찰하였다(표 12-3).

표 12-3. 연산수의 동적퍼센트

	Pascal	C	평균
옹근수상수	16	23	20
스칼라변수	58	53	55
배렬/구조체	26	24	25

Pasacal과 C프로그람들사이에 근사한 결과가 얻어 지는것은 대다수 참조가 단순한 스칼라변수들에서 일어 나는 경우이다. 더우기 80 %이상의 스칼라들은 틀내에서의 국부변수들이다. 뿐만아니라 배렬 및 구조체에 대한 참조는 보통 국부스칼라인 첨수 혹은 지적자로 선행참조를 요구한다. 결국 스칼라들에 대한 참조가 절대적인 자리를 차지한다.

Patterson 의 연구는 구성방식과 독립적으로 HLL 프로그람들의 동적인 동작을 조사하였다. 프로그람동작을 보다 깊이 있게 조사하자면 실제적인 구성방식에서 해야 한다. 일부 연구자들은 DEC-10 명령들을 동적으로 조사하여 평균 매 명령이 기억기에서는 0.5개의 연산수와 1.4개의 등록기들을 참조한다는것을 밝히였다[LUND77]. 이와 류사한 결과들이 S/370, PDP-11과 VAX에서 C, Pascal, FORTRAN프로그람들을 실행시키는 경우에 대하여서도 발표되였다[HCUK83]. 물론 이 결과들은 구성방식, 콤파일러와 깊은 관계를 가지지만 어디까지나 연산수호출빈도를 보여 주고 있는데 불과하다.

이상에서 고찰한 연구결과들은 수행되는 연산의 빈도가 높을수록 고속연산수호출에 알맞는 구성방식을 개발하는것이 중요하다는것을 말해 주고 있다. 특히 Patterson 의 연구결과는 최적화의 제1후보자가 국부스칼라변수들을 기억 및 호출하는 기구라는것을 보여 주고 있다.

3. 틀호출

틀호출 및 복귀가 HLL 프로그람의 중요한 측면이라는것은 이미 고찰하였다. 표 12-2를 보면 틀호출 및 복귀가 번역된 HLL프로그람에서 가장 많은 시간을 소비하는 연산으로 되고 있다. 따라서 이 연산을 효과적으로 실현하는 방법을 고찰하는것은 매우 중요한 문제로 된다. 이 문제에서는 두가지 측면 즉 틀에 관계되는 파라메터들과 변수들의 수 그리고 겹놓이는 깊이가 중요하다.

Tanenbaum 의 연구[TANE78]는 동적으로 호출된 틀들의 98 %가 6개의 인수보다 더 작은 인수들을 리용하며 동적으로 호출된 틀들의 92 %가 6 개의 국부스칼라변수보다

더 작은 변수들을 리용하였다는것을 밝히였다. 이와 류사한 연구결과들은 표 12-4 에서 보여 준바와 같이 Berkeley 의 RISC 연구집단에 의해서도 발표되였다[KATE83]. 이 결과들은 틀시동에 요구되는 단어들의 개수가 크지 않다는것을 보여 주고 있다. 그 이전에 보고된 연구결과들은 연산수참조의 큰 몫이 국부스칼라변수에 있다는것을 지적하였다. 이것은 연산수참조가 사실상 상대적으로 적은 변수들로 제한된다는것을 보여 주고 있다.

Berkeley 의 연구집단은 또한 HLL 프로그람들에서 틀호출 및 복귀현상을 고찰하였다. 연구집단은 대응하는 복귀렬에 뒤따르는 긴 비중단틀호출의 렬은 프로그람내에서 드물게 나타난다는것을 밝히였다. 오히려 이 연구집단은 프로그람이 틀호출깊이가 좁은 창문으로 제한되여 보존된다는것을 발견하였다.

표 12-4. 틀인수와 국부스칼라변수

실행된 틀 호출의 퍼센트	콤파일러, 해석프로그람,형설정기	작은 변수값 프로그람
>3 개의 인수	0-7%	0-5%
>5 개의 인수	0-3%	0%
>8 개의 인수 및 국부스칼라단어	1-20%	0-6%
>12 개의 인수 및 국부스칼라단어	1-6%	0-3%

이것이 제 4 절에서 론의하여 그림 4-29 에서 설명된것이다. 이 결과로부터 연산수참조가 매우 제한된다는 결론을 내릴수 있다.

4. 결론

여러 연구집단들은 HLL 에 밀접한 명령모임구성방식을 만들려는 시도가 결코 가장 효과적인 설계전략이 아니라고 결론한것 그리고 방금전에 소제목 3 에서 결론한것 등을 비롯하여 수많은 결과들을 발표하였다. 보통 일반적인 HLL 프로그람들에서 가장 많은 시간을 소비하는 연산특징들의 성능을 최적화하면 HLL 프로그람들을 잘 지원할수 있다.

여러 연구자들의 연구결과를 일반화하면 세가지 요소들이 RISC 구성방식을 특징 짓는다고 말할수 있다. 그것은 첫째로 많은 수의 등록기를 리용하거나 혹은 등록기사용을 최적화하도록 콤파일러를 리용하는것이다. 이것은 연산수참조의 최적화를 목적으로 한것이다. 방금전에 론의한 이와 관련한 연구들은 HLL 명령당 여러개의 연산수참조가 있으며 여기에서도 값주기명령문이 큰 몫을 차지한다는것을 보여 주고 있다. 이것은 곧 스칼라변수의 국부성 및 우세성과 결합되며 보다 많이 등록기참조를 할수록 기억기참조가 감소되여 성능을 높일수 있다는것을 말해 주고 있다.

둘째로는 명령관흐름설계에 깊은 관심을 돌리는것이다. 조건분기와 틀호출명령들이 실행시간에서 큰 몫을 차지하므로 간단한 명령관흐름은 비효과적이다. 이것은 미리 불러낼수는 있지만 절대로 실행되지는 않는 명령들의 몫을 크게 해야 한다는것을 말해 준다.

마지막으로 축소명령모임에 대한것이다. 이 점은 첫째, 둘째내용에 비하면 명백하지 못한 부분이며 앞으로 론의가 심화되면 명백해 질것이다.

제 2 절. 큰 등록기파일의 리용

제 1 절에서는 연산수고속호출의 필요성을 강조한 결과들을 개괄하였다. 또한 HLL 프로그람들에서 값주기명령문이 큰 몫을 차지하며 이 값주기명령문들중의 대다수는 A←B의 단순한 형태라는것을 보았다. HLL 명령문마다 하나이상의 연산수호출들도 있다. 이 결과들을 대다수 호출명령이 국부스칼라변수를 가진다는 사실과 결부시키면 등록기식기억기에 대한 믿음성이 더욱 커지게 된다.

등록기식기억기를 리용하는 리유는 이것이 주기억기나 캐쉬보다 더 빠른 가장 고속인 기억기이기때문이다. 등록기파일은 ALU 나 조종장치와 동일한 소편우에 있는 물리적으로 작은것이며 따라서 가장 빈번히 호출해야 할 연산수들은 등록기에 보존하여 등록기와 기억기사이 연산을 최소로 하는 전략이 필요하다.

여기에는 두가지 기본방법이 있다. 하나는 쏘프트웨어에 의한것이고 다른 하나는 하드웨어에 의한 방법이다. 쏘프트웨어방법은 콤파일러에 등록기를 최대로 리용하도록 의뢰하는 방법이다. 콤파일러는 등록기들을 주어 진 시간동안에 제일 많이 리용되는 변수들에 할당한다. 이 방법은 고급한 프로그람-해석알고리듬을 리용하는것을 전제로 한다. 하드웨어방법은 보다 많은 변수들을 오랜 시간동안 등록기에 유지할수 있도록 단순히 더 많은 등록기들을 리용하는 방법이다.

이 절에서는 하드웨어방법을 기본으로 고찰한다. 이 방법은 Berkeley 의 RISC 연구집단에 의해 제안되였으며 [PATT82 a] 첫 RISC 상품인 Pyramid 에 리용되였고 [RAGA83] 현재 일반 SPARC 구성방식에도 리용되고 있다.

1. 등록기창문

얼핏 보기에도 큰 규모의 등록기들을 리용하면 기억기호출에 대한 요구가 감소될것같다. 설계과제는 이 목적을 실현할수 있는 방향에서 등록기들을 구성하는것이다.

대다수의 연산수참조가 국부스칼라에 있으므로 한가지 명백한 방법은 전역변수로 예약된 몇개의 등록기들을 포함한 등록기들에 국부스칼라들을 기억하는것이다. 문제는 국부의 정의가 매 틀호출 및 복귀, 빈번히 일어 나는 연산들에 따라서 변하는것이다. 모든 호출에서 국부변수들은 등록기로부터 기억기로 옮겨 보관되여 이 등록기들을 호출한 프로그람이 다시 리용할수 있게 해야 한다. 뿐만아니라 파라메터들을 넘길 때에도 등록기들을 리용해야 한다. 또한 어미프로그람의 변수들을 등록기에 넣어야 하며 결과를 어미프로그람으로 넘겨야 한다.

이것은 이미 제 1 절에서 고찰한 다음의 두가지 서로 다른 결과들에 기초하면 된다. 첫째로, 일반적으로 틀은 몇개의 넘기기파라메터들과 국부변수들만을 리용한다(표 12-4). 둘째로, 틀시동의 깊이는 상대적으로 좁은 범위내에 영향을 미친다(그림 4-29). 이 속성들을 리용하자면 여러개의 작은 모임으로 된 등록기들을 리용해야 하며 여기서 매 작은 모임의 등록기들은 각이한 틀에 할당할수 있어야 한다. 틀호출은 처리장치가 각이한 등록기로 된 고정크기의 창문을 자동적으로 리용하도록 한다.

결국 린접틀들의 창문들이 파라메터를 통과시킬수 있게 서로 겹놓이게 된다. 이와 같은 개념을 그림 12-1 을 통하여 보여 주고 있다. 임의의 순간에는 오직 하나의 등록기창문만을 볼수 있으며 마치 이 등록기창문이 유일한 등록기모임인듯이 보고 주소화할수 있다(실례로 주소 0 부터 N-1 까지). 창문은 3 개의 고정크기령역으로 나누어 진다. 파라메터등록기들은 현재틀을 호출한 틀로부터 넘어 온 파라메터들을 가지며 또 거꾸로

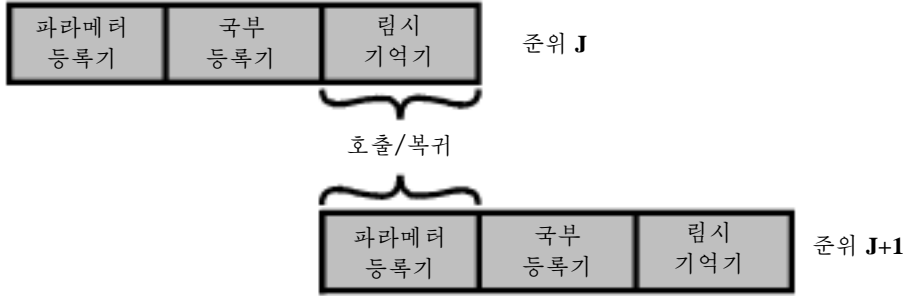



그림 12-1. 겹놓이는 등록기창문

넘겨 주어야 할 결과를 가지기도 한다. 국부등록기들은 콤파일러에 의해 할당될 때 국부변수로 리용된다. 림시등록기들은 다음의 보다 낮은 준위(현재 틀에서 호출한 틀)와 파라메터 및 결과를 교환하는데 리용된다. 한 준위에 있는 림시등록기들은 물리적으로 그 다음 낮은 준위에 있는 파라메터등록기들과 같이 볼수 있다. 이 겹놓임은 파라메터들을 자료의 실지이동이 없이 넘기도록 한다.

임의의 호출 및 복귀패턴들을 조종하자면 등록기창문의 수를 제한하지 말아야 한다. 대신 얼마 안되는 최신틀시동들만을 가지도록 등록기창문을 리용해야 한다. 이때 그 이전의 낡은 틀시동들은 기억기에 보관하거나 겹놓이는 깊이가 감소되면 후에 등록기창문에 복귀시켜야 한다. 결국 등록기파일의 실지 구성은 창문들이 겹놓이는 원형완충기와 같다.

이 구성을 그림 12-2에 보여 주었다.

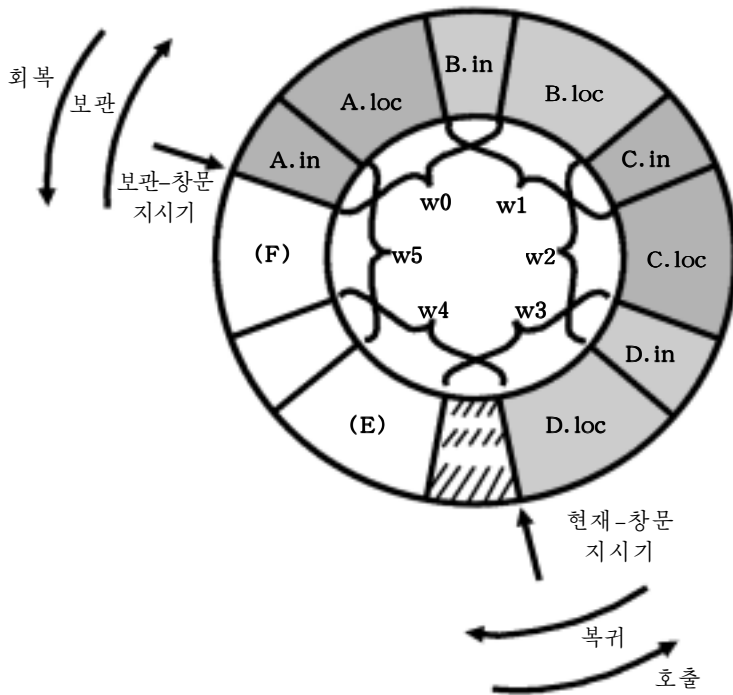



그림 12-2. 중첩된 창문들의 원형-완충

이 그림에서는 6 개의 창문을 가진 원형완충기를 보여 주고 있다. 완충기는 틀 D 가 능동(동작중)인 경우 4의 깊이로 채워 진다(A 는 B 를 호출, B 는 C 를 호출, C 는 D 를 호출). 현재창문지시기(CWP)는 현재능동틀의 창문을 가리킨다. 기계명령에 의한 등록기참조는 이 지시기가 실지 물리등록기를 가리키도록 변위된다. 보관창문지시기(SWP)는 기억기에 제일 마지막으로 보관된 창문을 식별해 준다.

이제 틀 D가 틀 E를 호출하였다면 E의 인수들은 D의 림시등록기들에 놓이게 되며(W3 과 W2 사이의 겹놓임) CWP 는 한 창문 전진하게 된다. 그다음 틀 E 가 틀 F 를 호출하면 이 호출은 현재의 완충기상태에는 부합될수 없다. 그것은 F 의 창문이 A 의 창문과 겹놓일수 있기때문이다. 만일 F 가 틀호출을 위해 준비한 그의 림시등록기들에 넣기를 시작한다면 A 의 파라메터등록기(A.in)에 겹쳐 써질것이다. 따라서 SWP 와 같아 질때까지 CWP 가 증가되면 새치기가 발생하여 A 의 창문이 보관된다. 실례로 F 의 시동에 뒤이어 B 가 A 로 되돌아 오면 CWP 는 감소되여 SWP 와 같아 지게 된다. 이것은 A 창문의 복귀로부터 초래되는 새치기를 발생시킨다.

이상과 같은 고찰로부터 N 창문등록기파일은 N-1 개의 틀시동만을 가질수 있다는것을 알수 있다. 이때 N 의 값은 크지 말아야 한다. 부록 4-1 에서 이미 언급한바와 같이 8 개의 창문을 가지면 보관 혹은 회복이 호출 혹은 복귀의 1 %정도로 된다[TAMI83]. Berkeley 의 RISC 콤퓨터는 매 창문이 16 개의 등록기로 된 8 개의 창문을 리용하고 있으며 Pyramid 콤퓨터는 매개가 32 개의 등록기인 16 개의 창문을 가지고 있다.

2. 전역변수

앞에서 서술한 창문방식은 등록기들에 국부스칼라변수들을 기억하는 효과적인 구성을 준다고 말할수 있다. 그러나 이 방식은 하나이상의 틀에서 호출하는 전역변수들을 기억해야 할 요구는 만족시키지 못한다. 전역변수에 대하여서는 다음과 같은 두가지 방식이 제안되고 있다. 첫째로 HLL 에서 전역변수로서 선언된 변수들이 콤파일러에 의해 기억기위치에 할당되고 이 변수들을 참조하는 모든 기계명령들이 기억기－참조연산수들을 리용하는 방식이다. 이 방식은 하드웨어나 쏘프트웨어(콤파일러)견지에서 볼 때 모두 간단한 방법이다. 그러나 전역변수의 호출빈도가 같을 때에는 비효과적인 방식으로 된다.

다른 하나의 방식은 처리장치내에 전역등록기모임을 공존시키는것이다. 이때 이 등록기들은 총적으로 고정되며 모든 틀들에 리용할수 있다. 단일번호달기방식(unified numbering scheme)은 명령형식을 간단히 하는데 리용한다. 실례로 0 부터 7 까지 번호가 붙은 등록기들에 대한 참조는 같은 전역등록기에 귀착시킬수 있으며 등록기 8 로부터 등록기 31 까지의 참조는 현재 창문에서 물리등록기들에 귀착되도록 변위시킬수 있다. 결국 등록기주소지정에서 분할을 알맞게 하자면 하드웨어부담을 증가시켜야 한다. 뿐만아니라 콤파일러가 어느 전역변수를 어느 등록기에 할당해야 하는가를 결정해야 한다.

3. 큰 등록기파일과 캐쉬의 비교

창문으로 구성된 등록기파일은 가장 널리 리용되는 변수들의 부분모임을 보존하기 위한 작고 빠른 완충기로서 작용한다. 이러한 관점에서 보면 등록기파일이 캐쉬와 매우 류사하게 작용한다고 볼수 있다. 이로부터 캐쉬와 등록기파일을 리용할 때 어느것이 더 간단하며 또 더 좋겠는가하는 의문이 생긴다.

표 12-5 는 이 두 장치들의 특성을 보여 주고 있다. 창문에 기초한 등록기파일은 최

신 N-1개의 틀시동에 필요한 모든 국부스칼라변수들을 가진다(드물게 일어 나는 창문의 자리넘침은 제외). 캐쉬는 마지막으로 리용되는 스칼라변수들중의 어느 하나를 선택하여 가진다. 등록기파일에는 늘 모든 국부스칼라변수들이 보존되여 있으므로 시간을 절약한다. 다른 한편 캐쉬는 동적으로 기억공간의 포화를 막으므로 기억공간을 보다 효과적으로 리용할수 있게 한다. 게다가 캐쉬는 일반적으로 명령 및 다른 형의 자료들을 포함하는 모든 기억기참조를 동일하게 취급한다. 따라서 이것들을 다른 령역으로 보관하는것은 캐쉬에서만 가능하며 등록기파일에서는 불가능하다.

표 12-5. 큰 등록기파일과 캐쉬구성의 특성

큰 등록기파일	캐쉬
모든 국부스칼라들	최신리용국부스칼라들
개별적인 변수들	기억기블로크
콤파일러할당전역변수들	최신리용전역변수들
틀겹놓임깊이에 기초한 보관/회복	캐쉬재배치알고리듬에 기초한 보관/복귀
등록기주소지정	기억기주소지정

등록기파일은 공간을 비효과적으로 리용하게 되는데 그것은 틀들이 자기에게 분배된 전체 창문공간을 다 리용하지 않기때문이다. 한편 캐쉬는 자료를 블로크단위로 읽어 들이는 또 하나의 비효과성을 가진다. 등록기파일은 리용중에 있는 변수들만을 가지지만 캐쉬는 리용되지 않는 일부 혹은 대부분의 자료를 블로크로 읽는다.

캐쉬는 국부변수들은 물론 전역변수들도 조종할수 있다. 보통 많은 전역스칼라들이 있지만 그것들중 몇개만이 자주 리용된다[KATE83]. 캐쉬는 동적으로 이 변수들을 발견하여 보유한다. 창문에 기초한 등록기파일이 전역등록기로 보충되면 그것 역시 일부 전역스칼라들을 가질수 있다. 그러나 콤파일러가 어느 전역변수가 자주 리용되는가를 결정하기는 힘들다.

등록기파일인 경우 등록기와 기억기사이 자료이동은 틀이 겹놓인 깊이에 의해 결정된다. 이 깊이는 보통 좁은 범위내에서 변동하므로 기억기리용은 상대적으로 빈번하지 못하다. 대다수 캐쉬들은 작은 크기의 묶음이 되도록 설정된다. 따라서 다른 자료 혹은 명령들이 자주 리용되는 변수들과 겹쳐 써질 위험성이 있다.

지금까지 론의한 내용에 기초하여 보아도 아직 큰 창문에 기초한 등록기파일과 캐쉬사이의 차이점은 명확하지 못하다. 그러나 등록기방식이 명백히 더 우월하며 여기서 캐쉬에 기초한 체계가 등록기방식보다 현저히 속도가 뜰것이라는것이 하나의 특징으로 된다. 이 차이는 두 방식들에서 주소지정에 간접적으로 소비되는 시간에서도 나타난다.

그림 12-3 은 이 차이를 그림으로 설명해 주고 있다. 창문에 기초한 등록기파일에서는 국부스칼라를 참조하기 위하여《가상》등록기번호와 창문번호를 리용한다. 이 번호들은 물리등록기들중의 어느 하나를 선택하는 상대적으로 단순한 해신기들을 통과한다. 캐쉬에서는 기억기위치를 참조하기 위하여 전체 주소모선에 기억기주소를 발생해야 한다. 이 연산의 복잡성은 주소지정방식에 관계된다. 하나의 묶음으로 된 캐쉬에서는 주소의 한 부분을 묶음크기와 같은 여러개의 단어들과 꼬리표들을 읽는데 리용한다. 이 부분을 제외한 주소의 다른 부분은 꼬리표와 비교되여 읽은 단어들중에서 어느 하나를 선택한다. 캐쉬가 등록기파일만큼 빠르다고 해도 호출시간이 상당히 더 길다는것은 명백한것이다.

따라서 성능의 견지에서 보면 국부스칼라인 경우는 창문에 기초한 등록기파일이 더 우월하다. 특히 명령의 경우에는 여기에 캐쉬를 추가하면 성능을 더 개선할수 있다.

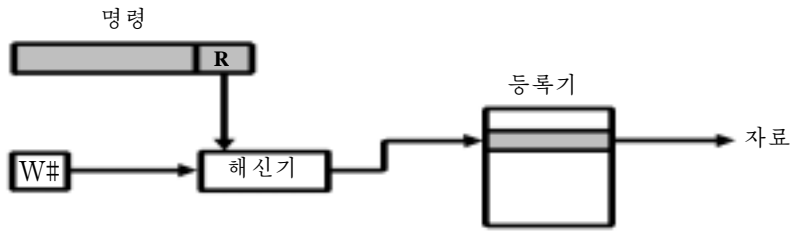
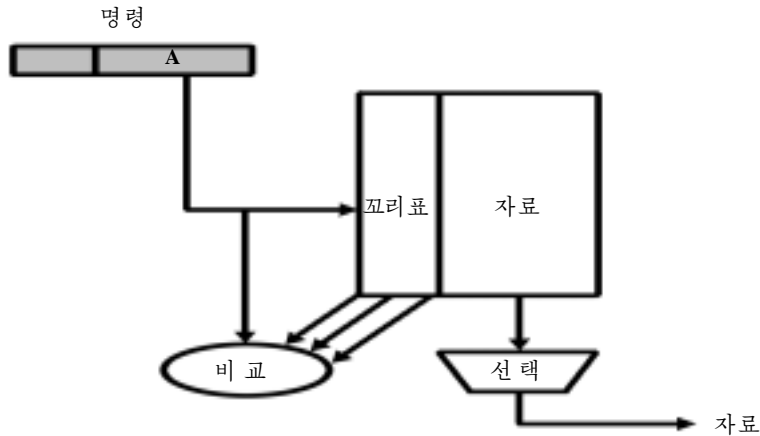



그림 12-3. 스칼라참조
ㄱ- 창문에 기초한 등록기파일, ㄴ- 캐쉬

제 3 절. 콤파일러에 기초한 등록기최적화

이제 작은 수 즉 16～32개의 등록기들만을 RISC 기계에서 리용할수 있다고 하자. 이 경우 등록기를 얼마나 최적으로 리용하는가 하는것은 콤파일러의 믿음성에 달려 있다. 물론 고급언어로 작성된 프로그람에서는 등록기들을 명시적으로 참조하지 않는다. 오히려 프로그람의 파라메터들을 기호로 표시한다. 콤파일러의 목적은 주기억기가 아니라 등록기에 가능한껏 많은 계산을 할수 있는 연산수를 대응시키며 또 넣기 및 기억연산을 최소로 하는것이다.

이를 위하여 일반적으로 취해 지는 방법은 다음과 같다. 등록기내에 존재하는 모든 프로그람의 파라메터는 기호 혹은 가상등록기에 배치되게 된다. 그다음 콤파일러는 이러한 무수히 많은 기호등록기들을 고정된 수의 실지등록기로 바꾼다. 기호등록기들은 그 사용이 겹치지 않지만 실지 등록기를 공유할수 있다. 프로그람의 특별한 부분에 처리해야 할 파라메터들이 실지 등록기보다 더 많은 경우에는 일부 파라메터들이 기억기위치에

배치되게 된다. 넣기 및 기억명령들은 연산을 위하여 림시로 등록기들에 파라메터들을 위치시키는데 리용된다.

최적화과제의 본질은 프로그람내의 임의의 주어 진 위치에서 어느 파라메터들이 등록기들에 넘겨 지게 될것인가를 결정하는것이다. RISC 콤파일러에서 가장 일반적으로 리용되고 있는것은 위상기하학의 원리로부터 받아 들인 기술인 도형색칠하기(graph coloring)이다[CHAI82, CHOW86, COUT86, CHOW90].

도형색칠하기는 다음과 같이 진행한다. 마디와 변으로 구성되는 하나의 그라프가 주어 지면 린접마디들이 서로 다른 색을 가지도록 마디들에 색을 할당하며 이 할당은 서로 다른 색의 수가 최소로 되도록 하는 원칙에서 한다. 이 문제를 다음과 같은 방법으로 콤파일러문제와 대응시킬수 있다. 우선 프로그람을 분석하여 등록기호상관계그라프를 그린다. 그라프의 마디들은 기호등록기들에 해당한다. 두개의 기호등록기가 같은 프로그람쪼각내에 존재한다면 등록기호상관계를 묘사하기 위해서 변으로 련결한다. 그다음 n 개의 색을 가지고 그라프를 색칠한다. 여기서 n 은 등록기의 수이다. 같은 색을 공유하는 마디들은 같은 등록기로 할당된다. 이 처리가 완전히 성공하지 못하면 색칠이 안된 마디들은 기억기에 배치하여 영향을 받은 파라메터들을 위한 공간을 만들어 넣기와 기억이 필요할 때 리용할수 있게 해야 한다.

그림 12-4 는 이러한 처리의 간단한 실례이다. 그림에서는 3 개의 실지등록기들로 번역되는 6 개의 기호등록기들을 가진 프로그람을 가정하고 있다. 그림 12-4 ㄱ는 매 기호등록기실지리용의 시간순서를 보여 주고 있으며 그림 12-4 ㄴ 는 등록기호상관계그라프(그늘진 부분과 사선친 부분은 색갈대신으로 리용하였다.)를 보여 주고 있다. 그림에서는 세가지 색갈로 색칠을 하였다. 그림에는 F 라는 하나의 기호등록기가 색을 칠하지 않은채로 남아 있는데 이것은 넣기 및 기억을 리용하여 처리해야 한다.

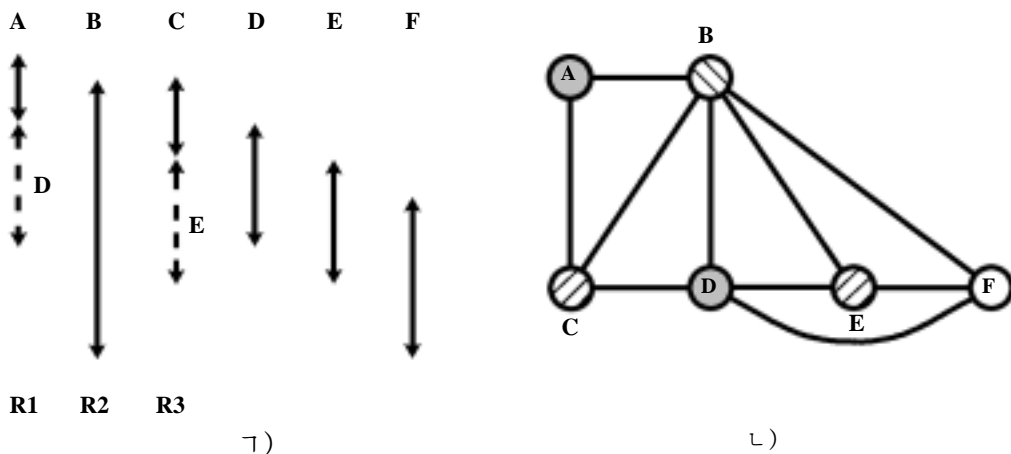



그림 12-4. 그라프색칠하기
ㄱ-실지 리용등록기의 시간순서, ㄴ- 등록기간섭그라프

일반적으로 큰 등록기모임의 리용과 콤파일러에 기초한 등록기최적화사이에는 합리적인 방안이 있다. [BRAD91a]는 Motorola 88000, MIPS R2000 과 류사한 특징을 가진 RISC 구성방식을 모형화한 연구결과에 대하여 보고하고 있다. 여기서는 등록기수를 16 으로부터 128 까지 변화시키면서 모든 일반등록기들의 리용과 옹근수 및 류점수리용에서의

등록기분할 등을 고찰하였다. 이 연구결과는 단순한 등록기최적화에서 64 개이상의 등록기를 리용하면 우점이 적어 진다는것을 보여 주었다. 합리적이며 고급한 등록기최적화기술인 경우 32 개이상의 등록기이면 가장 좋은 성능개선을 얻을수 있다. 결국 이 연구결과들은 작은 수의 읽기등록기들(례로 16)을 가지면 공유등록기구성을 가진 콤퓨터들이 분할구성을 가진 콤퓨터들보다 더 빨리 명령을 실행한다는것을 말해 준다. 이와 류사한 결과들이 여러 연구자들에 의해서도 발표되였으며[HUGU91] 이 결과들은 모두 기본적으로 작은 수의 등록기들을 리용한 최적화에 귀착된다.

제 4 절. 축소명령모임구성방식

이 절에서는 축소명령모임구성방식의 필요성과 일부 일반적인 특성에 대하여 고찰한다. 먼저 현대복합명령모임구성방식의 필요성에 대하여 론의한다.

1. 복합명령모임구성방식의 필요성

이미 앞에서 많은 명령들과 복잡한 명령들을 포함하는 보다 풍부한 명령모임에 대한 경향을 고찰하였다. 이러한 경향은 두가지 원리적인 리유 즉 콤파일러를 간단하게 하려는 요구와 성능을 개선하려는 요구로부터 나온것이다. 이 두가지 리유의 바탕에는 프로그람작성자들이 고급언어를 주로 리용하며 구성방식설계가들이 HLL 프로그람들을 더 잘 지원할수 있게 하는것이 있다.

이 장의 목적은 CISC 설계가들이 방향을 잘못 잡았다는것을 말하려고 하는것이 아니다. 기술이 끊임없이 발전하고 콤퓨터구성방식이 두개의 정연한 종류가 아니라 여러가지 종류로 존재하고 있으므로 흑백평가가 정확하지 않다. 따라서 뒤에서는 CISC 방법이 포함하고 있는 부족점들을 지적하고 RISC 지지자들의 립장에 대한 일부 정확한 리해를 설명한다.

앞에서 언급한 첫번째 론거 즉 콤파일러의 간단화는 명백하다. 콤파일러작성도구를 만들기 위한 과제는 매 HLL 명령문에 대하여 기계명령렬을 발생시키는것이다. 만일 HLL 명령문들과 비슷한 기계명령들이 있다면 이 과제는 간단히 해결된다. 여러 RISC 연구자들이 이 문제를 고찰하였다([HENN82], [RADI83], [PATT82b]). 그들은 콤파일러가 그 구조를 정확히 맞추는 경우를 반드시 찾아야 하므로 복잡한 기계명령의 개발이 매우 힘들다는것을 밝히였다. 발생한 코드를 그 크기가 최소로 되도록 최적화하기 위한 과제는 명령실행량을 줄이는것이며 관흐름처리의 성능제고는 복잡한 명령모임인 경우 훨씬 더 힘들다. 이 장의 맨 앞에서 이미 인용한 연구들은 번역된 프로그람에서 대다수의 명령들은 상대적으로 단순한 명령들이라는것을 보여 주었다.

두번째 론거의 요점은 CISC 가 더 작고, 더 고속인 프로그람을 산생할것이라는 기대이다. 이 주장의 두가지 측면 즉 프로그람이 보다 작으며 프로그람이 보다 고속으로 실행될것이라는 주장을 고찰해 보자.

우선 보다 작은 프로그람에는 두가지 우점이 있다. 하나는 적은 기억기를 가지므로 그 자원이 절약된다는것이다. 오늘날 기억기비용이 매우 눅어 진것을 고려하면 이 우점은 더는 지속되지 못할것이라고 본다. 다른 하나의 중요한 우점은 보다 작은 프로그람이 성능을 개선한다는것인데 그 리유는 다음과 같다. 첫째로, 보다 작은 명령들은 불러 들여야 할 보다 작은 수의 명령바이트들을 의미하며 둘째로, 페지화환경에서 보다 작은 프로그람은 페지결함을 감소시키면서 보다 적은 페지를 차지한다.

이와 같은 론거에서 문제는 CISC 프로그람이 대응하는 RISC 프로그람보다 더 작을것이라는 추측이 맞지 않는것이다. 기호기계어로 표시된 CISC 프로그람은 더 작은 명령들을 가지지만 그것이 차지하는 기억기비트수는 현저히 더 작아 지지 않는다. 표 12-6 에는 축소명령모임구성방식의 RISC I 를 포함하여 여러가지 콤퓨터들에서 번역된 C 프로그람의 크기를 비교한 결과들을 보여 주고 있다. 이 표로부터 RISC 프로그람에서보다 CISC 프로그람을 리용하는 경우 약간의 절약이 있거나 거의 절약이 없다는것을 알수 있다. 특히 흥미 있는것은 PDP-11 보다 훨씬 더 많은 복잡한 명령모임을 가지는 VAX 가 PDP-11 보다는 매우 적은 기억기령역절약을 가져 온다는것이다. 이와 같은 결과들은 IBM 연구자들에 의해 확증되였다[RADI83]. 그들은 IBM 801 이 IBM S/370 코드크기의 0.9 배정도의 코드를 가진다는것을 밝혔으며 이 연구는 PL/I 프로그람의 명령모임을 가지고 진행하였다.

이러한 놀라운 결과들에는 여러가지 원인들이 있을수 있다. 이미 CISC 의 콤파일러들이 보다 단순한 명령들에 알맞게 되여 있으므로 복잡한 명령들의 간결성이 드물게 나타난다는것을 보았다. 또한 CISC 에서는 보다 많은 명령들이 있으므로 긴 조작코드가 필요하며 이것은 긴 명령을 낳는다. 결국 RISC 는 기억기참조가 아니라 등록기참조를 강화하기 쉬우며 이것은 보다 적은 비트들을 요구한다.

이상과 같은 고찰로부터 CISC 가 더 작은 프로그람을 산생할것이라는 기대는 실현될수 없다. 복잡한 명령모임들에서의 두번째 자극인자는 명령실행이 더 빠를것이라는것이였다. 이것은 복잡한 HLL 연산이 보다 원시적인 명령들의 렬에서보다는 단일한 기계명령에서 더 빨리 실행될것이라는 느낌을 준다. 그러나 보다 단순한 명령들의 리용을 지향하므로 이것은 그렇게 될수는 없다. 전체 조종장치가 더 복잡하게 구성되므로 마이크로프로그람조종기억을 보다 풍부한 명령모임에 맞게 더 크게 구성해야 한다.

표 12-6. RISC I 와의 상대적인 코드크기

	11 개의 C 프로그람 [PATT82a]	**12 개의 C 프로그람** [KATE83]	**5 개의 C 프로그람** [HEAT84]
RISC I	1.0	1.0	1.0
VAX-11/780	0.8	0.67	
M68000	0.9		0.9
Z8002	1.2		1.12
PDP-11/70	0.9	0.7	

일부 연구자들은 복잡한 명령실행에서 속도상승이 고속조종기억기에 이 명령들을 상주시키는데 있는것이 아니라 복잡한 기계명령들의 능력에 있다는것을 밝히였다[RADI83]. 결과적으로 조종기억기는 명령캐쉬로서 작용한다. 따라서 하드웨어설계가들은 어느 부분루틴 혹은 어느 함수들이 가장 빈번히 리용되는가를 결정하고 이것들을 마이크로코드로 실현하여 조종기억기에 할당해야 한다. 이 결과는 아직 불충분하다. S/390 체계에서 번역 및 확장된 정확도 류점수나누기와 같은 명령들은 고속기억기에 상주시키며 한편 틀호출을 설정하거나 새치기조종기를 초기화할 때 동반되는 명령렬들은 보다 속도가 뜬 주기억기에 위치시킨다.

결국 매우 복잡한 명령모임을 지향하는것이 합리적이라고 보기는 힘들다. 이러한 사실들은 많은 연구집단들이 이와는 반대의 길을 모색하게 하고 있다.

2. 축소명령모임구성방식의 특성

축소명령모임구성방식에는 여러가지가 있지만 이것들은 모두 공통적으로 다음과 같은 특성을 가지고 있다.

- 주기당 하나의 명령
- 등록기 대 등록기연산
- 단순한 주소지정방식
- 단순한 명령형식

이 특성들을 차례로 고찰해 보자.

첫번째 특성은 **기계주기당 하나의 기계명령**이 있다는것이다. 여기서 **기계주기**는 등록기로부터 2 개의 연산수를 불러 내여 ALU 연산을 수행하고 그 결과를 등록기에 기억하는데 걸리는 시간으로 정의된다. 따라서 RISC 기계명령들은 CISC 콤퓨터들에서의 마이크로명령보다는 고급하지 못하지만 그만큼 빨리 실행된다. 간단히 하나의 기계주기를 생각하면 이때 마이크로코드는 거의나 필요 없게 된다. 이러한 명령들은 콤퓨터들에서의 다른 기계명령들과 비교해 볼 때 더 빨리 실행될수 있다. 그것은 명령을 실행하면서 마이크로프로그람조종기억기의 호출이 필요 없기때문이다.

두번째 특성은 대부분의 연산이 기억기를 호출하는데 단순히 넣기 및 기억연산만을 리용하는 **등록기 대 등록기**연산이라는것이다. 이 특징은 명령모임을 간단하게 하며 따라서 조종장치도 단순하게 한다. 실례로 RISC 명령모임은 하나 혹은 두개의 ADD 명령 즉 옹근수더하기와 자리올림을 고려한 더하기명령을 가지고 있지만 VAX 는 25 개의 각이한 더하기명령을 가지고 있다. 설계특징은 또한 이러한 구성방식이 등록기를 최적으로 리용하게 하며 자주 호출되는 연산수들을 고속기억기에 보유한다는것이다.

이와 같은 등록기 대 등록기연산의 강화는 RISC 설계에서만 찾아 볼수 있는 유일한 것이다. 다른 현대콤퓨터들에서도 이러한 명령들을 가지고 있지만 이외에도 기억기 대 기억기연산, 혼합된 기억기 및 등록기연산들도 포함하고 있다. 이러한 방법들에 대한 비교는 RISC 가 출현하기전인 1970 년대에 진행되였다. 그림 12-5 에서는 기억기 대 기억기연산과 등록기 대 등록기연산을 비교하고 있다. 여기서는 가상적인 구성방식에 대하여 프로그람크기와 기억기호출비트수를 평가하였다. 이 과정에 어떤 연구자는 미래의 구성방식이 등록기들을 전혀 포함하지 않을것이라는것을 제기하였다[MYER78]. 이때 사람들은 이 사람이 과연 그때 528 개의 등록기를 가진 Pyramid 라는 이름으로 된 RISC 콤퓨터가 이미 나와 있다는것을 생각하였을가 하고 의심스럽게 생각하였다.

이들은 우선 작은 수의 국부스칼라들에 대한 호출이 매우 빈번하다는것을 인식하지 못하였으며 큰 묶음의 등록기 혹은 최적화된 콤파일러를 가지게 되면 대다수 연산수를 오랜 시간 등록기에 유지할수 있다는것을 몰랐다. 이것을 고려한 비교를 그림 12-5 ㄴ 에 보여 주었다.

세번째 특성은 **단순한 주소지정방식**의 리용이다. 거의 모든 RISC 명령들은 단순한 등록기주소지정방식을 리용한다. 변위나 PC 상대와 같은 여러가지 추가적인 방식들도 리용할수 있다. 이보다 더 복잡한 방식들은 단순한 방식들을 쏘프트웨어로 합성하여 얻을수 있다. 이 특성은 명령모임과 조종장치를 간단하게 한다.

마지막 네번째 특성은 **단순한 명령형식**의 리용이다. 일반적으로 RISC 에서는 하나 혹은 몇개의 형식들만 리용된다. 명령길이는 고정되며 단어경계로 배치된다. 마당의 위치 특히 조작코드는 고정이다. 이 특징은 여러가지 리익을 가져다 준다. 고정된 마당을 가지

게 되면 조작코드해신과 등록기연산수호출이 동시에 일어 날수 있다. 단순화된 형식은 또한 조종장치를 간단하게 한다. 명령꺼내기는 단어길이단위로 진행되므로 최적화된다. 단어단위의 값주기는 또한 단일명령이 페지경계를 가로 지르지 않는다는것을 의미한다.

8	16	16	16
Add	**B**	**C**	**A**

기억기 대 기억기
I = 56, D = 96, M = 152

8	4	16	
Load	**rB**	**B**	
Load	**rC**	**B**	
Add	**rA**	**rB**	**rC**
Store	**rA**	**A**	

등록기 대 기억기

ㄱ)

8	16	16	16
Add	**B**	**C**	**A**
Add	**A**	**C**	**B**
Sub	**B**	**D**	**D**

기억기 대 기억기
I = 168, D = 288, M = 456

8	4	4	4
Add	**rA**	**rB**	**rC**
Add	**rB**	**rA**	**rC**
Sub	**rD**	**rD**	**rB**

등록기 대 기억기
I=104, D=96, M=200

ㄴ)

I – 확장명령의 크기
D – 확장자료의 크기
M = I+D – 총기억기 통신량

그림 12-5. 두가지 등록기 대 등록기와 기억기 대 기억기방법의 비교
ㄱ- A ← B+C, ㄴ-A ← B+C; B ← A+C; D ← D-B

이상에서 고찰한 특성들은 RISC 방식의 잠재적인 우점을 모두 결정한다. 이 우점들은 기본적으로 두 부류로 나누어 볼수 있는데 하나는 성능에 관한 우점이고 다른 하나는 VLSI 실현과 관련한 우점이다.

성능에 대하여 먼저 고찰해 보자. 우선 보다 효과적이며 최적화된 콤파일어를 개발할수 있다. 명령들이 원시적일수록 고리의 밖으로 함수들을 이동하거나 효과성 있게 코드를 재구성할수 있으며 등록기를 최대로 리용할수 있다. 지어 번역시간에 복잡한 명령들의 일부를 계산하는것도 가능하다. 실례로 s/390 의 문자이동 MVC 명령은 한 위치로부터 다른 위치로 문자렬을 이동한다. 이 명령이 실행될 때마다 이동은 문자렬의 길이에만 관계되며 어느 방향에서 위치들의 겹쌓임이 있는가 또 일치(정렬)특성이 있는가에는 무관계하다. 대다수 경우에 이러한 내용들은 번역시간에 모두 알게 된다. 따라서 콤파일러는 이 기능에 대한 최적인 원시명령렬을 산생할수 있다.

다음으로는 콤파일러에 의해 생겨 난 대다수 명령들이 상대적으로 단순하다는것이다. 매우 적은 마이크로코드를 리용하거나 혹은 마이크로코드를 전혀 리용하지 않는 조종장치는 이렇게 얻어 진 명령들을 위하여 특별히 만든것으로서 CISC 조종장치보다 이 명령렬을 더 빠르게 실행할수 있다.

성능과 관련한 세번째 우점은 명령관흐름처리를 리용한다는것이다. RISC 연구자들은 명령관흐름처리기술을 축소명령모임인 경우에 훨씬 더 효과적으로 적용할수 있다는것을

밝히였다. 이 문제에 대하여 곧 상세히 고찰한다.

다소 의의가 적다고 볼수 있는 마지막우점은 새치기들을 요소적인 연산들사이에서 검사하므로 RISC 프로그람들이 새치기에 대한 응답성이 더 좋다는것이다. 복잡한 명령을 가진 구성방식들은 명령한계로 새치기들을 제한하거나 혹은 특별한 새치기가능점들을 정의하여 명령을 재시동하는 기구를 구비해야 한다.

축소명령모임구성방식에서 성능이 개선된 상황은 그 증명이 힘들다. 이에 대한 여러가지 연구들이 진행되였지만 그것은 어디까지나 비교할만한 기술과 능력을 가진 콤퓨터들에서 수행된것들이 아니다. 더우기 이 대다수 연구들은 축소명령모임의 영향과 큰 등록기파일의 영향을 분리시켜 고찰하지 못하였다.

다음으로 보다 명백한 두번째 RISC의 우점은 VLSI의 실현과 관련된다. VLSI를 리용하는 경우 처리장치설계와 그 실현은 근본적으로 달라 진다. IBM S/390과 VAX와 같은 전통적인 처리장치들은 표준화된 SSI와 MSI 패키지를 가지는 하나이상의 인쇄회로판들로 구성된다. LSI 와 VLSI 의 출현으로 단일소편우에 하나의 처리장치를 놓을수 있게 되였다. 단일소편처리장치에서는 RISC 전략과 관련하여 두가지 문제점들이 있다. 첫 문제점은 그의 성능이다. 보통 단일소편우에서의 지연은 소편들사이의 지연에 비해 훨씬 더 작다. 따라서 한 소편으로 처리장치를 실현하면 자주 발생하는 명령들에 매우 효과적이다. 실지로 단순명령들과 국부스칼라들에 대한 호출은 가장 많이, 자주 일어 나게 된다. Berkeley 의 RISC 소편들은 이러한 고찰에 기초하여 설계된것들이다. 한편 전형적인 단일소편극소형처리장치들은 그의 령역의 약 절반을 마이크로코드조종기억기로 쓰지만 RISC Ⅰ소편은 그의 령역의 약 8 %만을 조종장치에 쓰고 있다[SHER84].

VLSI와 관련한 두번째 문제점은 설계와 실현시간이다. VLSI 처리장치는 보통 개발하기가 힘들다. 설계가들은 SSI나 MSI부품을 리용하지 않고 회로설계와 지면배치, 장치준위에서의 모형화 등을 진행해야 한다. 축소명령모임구성방식을 리용하면 이 처리를 표 12-7 에서 보여 준바와 같이 매우 쉽게 할수 있다[FITZ81]. RISC 소편의 성능이 비교대상으로 되는 CISC 극소형처리장치들의 성능과 같게 되면 그때에는 RISC 방식의 우점이 명백해 진다.

표 12-7. 일부 극소형처리장치들의 설계와 지면배치로력수

CPU	3 극소자(× 1000)	설계(명-월)	지면배치(명-월)
RISC Ⅰ	43	15	12
RISC Ⅱ	41	18	12
M68000	68	100	70
Z8000	18	60	70
Intel iAPx-432	110	170	90

3. CISC에 비한 RISC의 특성

RISC 기계들이 출현한 이후 그의 실현과 인식에서는 큰 전진이 있었다. 그 성장은 다음과 같다.

(1) RISC 설계는 일부 CISC 특징들을 포함하면 더 효과적이다.

(2) CISC 설계는 일부 RISC 특징들을 포함하면 더 효과적이다.

이와 같은것을 고려하여 PowerPC로 대표되는 최신 RISC 설계에서는 순수 RISC 방식을

더이상 리용하지 않고 있으며 Pentium II로 대표되는 최신 CISC설계에서는 일부 RISC 특성들을 결합시키고 있다.

일부 연구자들이 제기한 흥미 있는 비교결과들은 이 문제에서 일정한 견해나 주견을 가지게 한다[MASH95]. 표 12-8 은 여러개의 처리장치들을 목록화하여 그것들의 여러가지 특성들을 비교하고 있다.

표 12-8. 일부 처리장치들의 특성

처리장치	명령크기의 수	바이트로 표시된 명령크기의 수	주소지정 방식의 수	간접 주소 지정	산수연산과 결합된 넣기 및 기억	기억기 연산수의 최대수	허용된 비정렬 주소지정	MML 리용의 최대수	옹근수 등록기주소 지정자를 위한 비트수	FP 등록기 지정자에 해당하는 비트수
AMD29000	1	4	1	없다	없다	1	없다	1	8	3 *
MIPS R2000	1	4	1	없다	없다	1	없다	1	5	4
SPARC	1	4	2	없다	없다	1	없다	1	5	4
MC68000	1	4	3	없다	없다	1	없다	1	5	4
HPPA	1	4	10	없다	없다	1	없다	1	5	4
IBM RT/PC	2 *	4	1	없다	없다	1	없다	1	4 *	3 *
IBM RS/6000	1	4	4	없다	없다	1	있다	1	5	5
Intel i860	1	4	4	없다	없다	1	없다	1	5	4
IBM 3090	4	8	2 #	없다#	있다	2	있다	4	4	2
Intel 80486	12	12	15	없다#	있다	2	있다	4	3	3
NSC 32016	21	21	23	있다	있다	2	있다	4	3	3
MC68040	11	22	44	있다	있다	2	있다	8	4	3
VAX	56	56	22	있다	있다	6	있다	24	4	0
Clipper	4 *	8 *	9 *	없다	없다	1	0	2	4 *	3 *
Intel 80960	2 *	8 *	9 *	없다	없다	1	있다*	-	5	3 *

* : 이 특성에 맞지 않는 RISC
\# : 이 특성에 맞지 않는 CISC

표에서의 비교를 위하여 다음과 같은 대표적인 RISC방식을 생각한다.

1) 단일명령크기
2) 그 크기는 대표적으로 4byte이다.
3) 대표적으로 5 보다 작은 수의 자료주소지정방식. 이 파라메터는 정확히 지키기가 힘들다. 등록기와 문자방식은 표에서 고려하지 않았으며 각이한 변위크기를 가진 여러가지 형식들을 따로 계산하였다.
4) 기억기에서 다른 연산수의 주소를 얻기 위해 임의의 기억기호출을 요구하는 간접주소지정방식은 없다.
5) 넣기 및 기억을 산수연산과 결합하는 연산은 없다(실례로 기억기로부터의 더하기, 기억기에 더하기).
6) 명령당 하나이상의 기억기주소연산수는 없다.
7) 넣기 및 기억연산을 위한 임의의 자료정렬은 지원하지 않는다.
8) 한 명령에서는 자료주소를 위한 기억기관리단위(MMU)의 최대수리용
9) 5혹은 그이상의 옹근수등록기지정자를 위한 비트수. 이것은 적어도 32개의 옹근수등록기들을 한번에 명시적으로 참조할수 있다는것을 의미한다.

10) 4 혹은 그이상의 류점수등록기지정자의 비트수. 이것은 적어도 16개의 류점수등록기들을 동시에 명시적으로 참조할수 있다는것을 의미한다.

항목 1 부터 3까지는 명령해신복잡성을 나타낸다. 항목 4부터 8까지는 특별히 가상기억기요구가 있는 조건에서 관흐름처리의 용이성 혹은 곤난성을 제기하고 있다. 항목 9 부터 10까지는 콤파일러를 리용하는 능력과 관련된다.

표에서 첫 8개의 처리장치들은 명확히 RISC 구성방식의 처리장치들이며 다음 4개는 CISC 구성방식의 처리장치이고 마지막 두개의 처리장치는 실지로 CISC 의 특성을 많이 가지고 있는 RISC 처리장치라고 볼수 있다.

제 5 절. RISC 의 관흐름처리

1. 규칙명령을 가진 관흐름처리

제 11 장 제 4 절에서 고찰한바와 같이 명령관흐름처리는 보통 성능강화에 리용된다. 이제 RISC 구성방식의 상황에서 이것을 고찰해 보자. RISC 에서는 대다수 명령들이 등록기 대 등록기명령이며 명령주기는 다음과 같은 두 단계를 가진다.

- I : 명령꺼내기
- E : 실행. 등록기에로의 입구와 등록기로부터의 출구를 가지고 ALU 연산을 수행한다.

넣기 및 기억연산인 경우 다음과 같은 세가지 단계가 필요하다.

- I : 명령꺼내기
- E : 실행.기억기주소계산
- D : 기억.등록기 대 기억기 혹은 기억기 대 등록기연산

그림 12-6 ㄱ는 관흐름처리를 리용하지 않는 명령렬의 사간관계를 보여 주고 있다. 이것은 명백히 쓸모 없는 처리라는것을 알수 있다. 매우 다양한 관흐름처리라고 해도 성능을 어느 정도밖에 개선할수 없다. 그림 12-6 ㄴ는 두방향관흐름처리방식을 보여 주고 있다. 두방향관흐름처리에서는 두개의 서로 다른 명령들이 I 및 E 단계에서 동시에 수행된다. 이 방식은 관흐름처리를 진행하지 않은 방식 즉 직렬방식의 실행속도보다 2 배의 속도상승을 가져 올수 있다. 그러나 다음과 같은 두가지 리유로 최대속도상승을 달성하기 힘들다. 그것은 첫째로, 단일포구기억기를 리용하며 단계마다 하나의 기억기호출만이 가능하다고 가정하였기때문이다. 이것은 일부 명령들에서 대기상태를 삽입할것을 요구한다. 두번째 리유는 분기명령들이 련속실행흐름을 중단하기때문이다. 최소순환성을 가지고 이것을 순응시키기 위하여 NOOP 명령이 콤파일러 혹은 아쎔블러에 의한 명령흐름에 삽입될수 있다.

관흐름처리는 한 단계당 두개의 기억기호출이 가능하면 그 성능을 더욱 개선할수 있다. 이 원리에 기초하여 얻어 진 명령렬을 그림 12-6 ㄷ에 보여 주고 있다. 여기서는 3 개의 명령을 중복시켜 그 성능을 3 배로 높이고 있다. 또한 분기명령들은 그 최대값까지 속도상승을 떨구지 않지만 떨구는것만은 사실이다. 뿐만아니라 자료의 상관성(의존성) 도 속도상승에 영향을 준다. 만일 한 명령이 실행명령에 의해 교체되는 연산수를 요구한다면 지연이 생긴다. 이 지연은 NOOP 명령에 의해 맞출수 있다.

이제까지 론의한 관흐름처리는 3 개의 단계가 근사적으로 같은 존속시간에 있는 경

우에 가장 효과적이다. 그림 12-6에서 E단계는 보통 ALU연산을 동반하므로 다른 단계보다 더 시간이 오래 걸린다. 이 경우에는 이 단계를 다음과 같은 두개의 부분단계로 분할할수 있다.

- E1: 등록기파일읽기
- E2: ALU연산과 등록기쓰기

명령모임이 단순하고 규칙적이므로 3 혹은 4 개의 단계로 맞추는 설계를 쉽게 할수 있다. 그림 12-6 ㄹ는 네방향관흐름의 결과를 보여 주고 있다. 이 관흐름에서는 단번에 4개의 명령까지 처리할수 있으며 최대로 가능한 속도상승은 직렬방식에 비해 4 배로 된다. 여기서도 자료의 의존성 및 분기명령을 고려한 NOOP 명령의 리용이 지연을 발생시킨다는것을 다시금 강조한다.

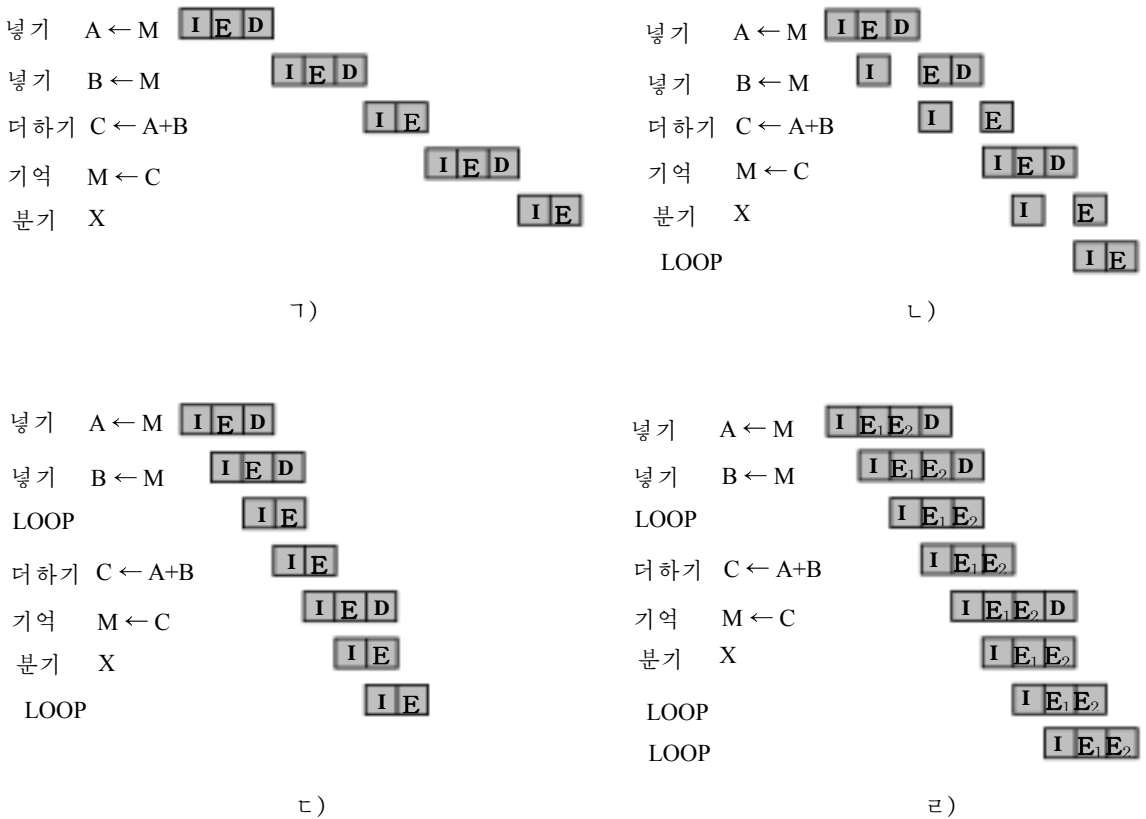



그림 12-6. 관흐름조종의 효과
ㄱ- 순차실행, ㄴ- 2단관흐름동기, ㄷ-3단관흐름동기, ㄹ-4단관흐름동기

2. 관흐름처리의 최적화

RISC 명령은 단순하면서도 규칙적인 명령이므로 여기에 관흐름처리방식을 효과적으로 적용할수 있다. 보통 명령이 실행되는 동안에 얼마 안되는 변동이 있을수 있으나 이

것을 반영하여 관흐름이 재구성된다. 그러나 자료 및 분기의 의존성 혹은 상관성으로 인하여 총체적인 실행속도는 감소하게 된다.

이 의존성에 대한 보상으로서 코드재구성기술이 개발되여 리용되고 있다. 이것을 보기 위하여 우선 분기처리명령을 고찰해 보자. 관흐름의 효과성을 증가시키는 한가지 방법인 지연분기는 분기명령에 뒤따르는 명령이 실행된후에야 분기명령을 실행하게 한다. 분기명령에서 분기를 즉시 뒤따르는 명령위치까지를 지연틈이라고 한다. 이 이상한 틈을 표 12-9 에서 설명하였다. 《정상분기》라는 표식이 붙은 렬에 있는것은 정상기호명령기계어프로그람이다. 이 틀에서 보면 102가 실행된후에 실행되여야 할 다음명령은 105이다. 관흐름을 규칙화하기 위하여 이 분기명령다음에 NOOP 명령을 삽입한다. 이것이 지연분기이다. 그러나 성능증가는 101 과 102 의 명령들을 서로 바꿀 때 일어 난다. 그림 12-7 은 이 결과를 보여 주고 있다.

표 12-9. 정상 및 지연분기

주 소	정상분기		지연분기		최적지연분기	
100	LOAD	X, A	LOAD	X, A	LOAD	X, A
101	ADD	1, A	ADD	1, A	JUMP	105
102	JUMP	105	JUMP	106	ADD	1, A
103	ADD	A, B	NOOP		ADD	A, B
104	SUB	C, B	ADD	A, B	ADD	A, B
105	STORE	A, Z	SUB	C, B	STORE	A, Z
106			STORE	A, Z		

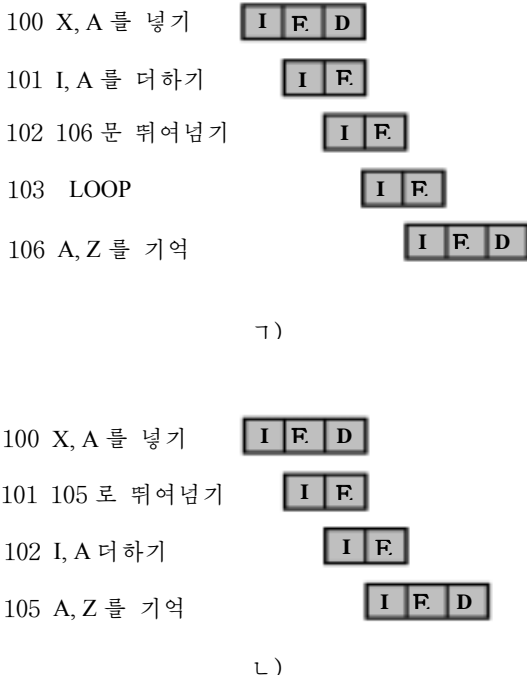



그림 12-7. 지연분기의 리용
ㄱ-삽입된 명령, ㄴ-예약된 명령

그림에서 보면 JUMP 명령을 ADD 명령보다 먼저 불러 낸다. 그러나 명심해야 할 것은 JUMP명령의 실행으로 프로그람계수기의 내용이 교체되기전에 ADD 명령을 불러 낸다는것이다. 결국 프로그람의 원래 의미는 보존된다.

이러한 명령의 교체는 무조건분기, 호출과 복귀 등에서 성과적으로 리용할수 있다. 조건분기인 경우에는 이러한 틀이 맹목적으로 적용되지 않는다. 분기를 위해 검사되는 조건이 선행명령으로 즉시 바꾸어 지는 경우에는 콤파일러가 교체를 그만 두어야 하며 대신 NOOP 를 삽입해야 한다. 이와 반대인 경우에는 콤파일러가 분기명령 다음에 목적하는 명령을 삽입할수 있다. Berkeley RISC 와 IBM 801 체계에서의 경험은 대다수 조건분기명령들을 이와 같은 형식으로 최적화할수 있다는것을 보여 주고 있다([PATT82 a], [RADI83]).

이와 류사한 지연넣기라고 하는 방법을 LOAD 명령에도 리용할수 있다. LOAD 명

령에서 넣기목표로 되는 등록기는 처리장치에 의해 밀페되여 있다. 처리장치는 이 등록기를 요구하는 명령을 만날 때까지 명령흐름의 실행을 계속하며 넣기가 없는 한 이 등록기는 놀고 있다. 콤파일러가 넣기명령의 관흐름에 있는 경우 쓸모 있는 작업을 수행하도록 명령을 재배치할수 있는 능력을 가지면 효과성이 증가된다.

마지막으로 강조하고 싶은것은 명령관흐름설계를 그 체계에 적용되는 다른 최적화기술과 분리시켜 별도로 하지 말아야 한다는것이다. 실례로 관흐름의 명령실행일정맞추기와 동적인 등록기할당은 가장 큰 효과성을 얻을수 있도록 명령관흐름설계와 함께 고찰해야 한다[BRAD91b].

제 6 절. MIPS R4000

처음으로 상품화되여 리용한 처리장치소편들중에는 MIPS 기술회사가 개발한것도 있다. 이 절에서는 MIPS R4000 에 대하여 고찰한다. MIPS R4000 은 실제상 이전의 MIPS 설계(R2000, R3000 과 R6000 설계)와 같은 구성방식, 같은 명령모임을 가진다. 가장 중요한 차이는 R4000 이 모든 내부 및 외부자료경로, 주소지정, 등록기 그리고 ALU 에 대하여 32bit 가 아니라 64bit 를 리용한다는것이다.

64bit 를 리용하면 32bit 구성방식에 비해 여러가지 우점이 있다. 즉 조작체계에 충분히 큰 주소공간을 할당하므로 가상기억기에 직접 Tbyte 이상의 파일을 배치할수 있으며 따라서 기억기호출이 쉽다. 오늘날 일반적으로 리용되고 있는 1Gbyte 이상의 디스크구동기를 가지게 되면 32bit 콤퓨터인 경우 4 Gbyte 주소공간이 극한값으로 된다. 또한 64bit 용량은 R4000 이 IEEE 의 단정확도류점수와 같은 자료나 한번의 동작에서 8 개 문자까지의 문자렬자료를 처리할수 있게 한다.

R4000 처리장치소편은 두개의 부분으로 나누어 져 있다. 하나는 CPU 이며 다른 하나는 기억기관리를 위한 협동처리장치이다. 이 처리장치는 매우 단순한 구성방식으로 되여 있다. 처리장치의 목적은 성능을 강화하는 론리를 리용할수 있는 공간(실례로 완전한 기억기관리단위)을 조성하면서 명령실행론리가 단순한 체계를 설계하는것이였다.

처리장치는 32 개의 64bit 등록기를 가지고 있다. 또한 128Kbyte 까지의 높은 속도 캐쉬를 장비하며 이 128kbyte 의 용량은 명령과 자료에 각각 절반씩 배당된다. 상대적으로 큰 캐쉬(IBM 3090 은 128～256Kbyte 의 캐쉬를 구비하고 있다.)는 주기억기모선에 넣지 않거나 창문론리를 가진 큰 등록기파일을 요구하지 않고도 처리장치에 국부적으로 큰 묶음의 프로그람코드와 자료를 유지할수 있게 한다.

1. 명령모임

표 12-10 에는 모든 MIPS 계렬처리장치들의 기본명령모임을 보여 주었다. 또한 표 12-11 에는 R4000 에서 실현된 추가적인 명령들을 보여 주었다. 모든 처리장치의 명령들은 단일한 32bit 단어형식으로 부호화된다. 모든 자료연산은 등록기 대 등록기연산이며 기억기참조만이 순수한 넣기 및 기억연산들이다.

R4000 에서는 조건코드를 리용하지 않는다. 어떤 명령이 조건을 발생하면 그에 대응하는 기발들이 일반등록기에 기억된다. 이렇게 하면 조건코드들이 관흐름처리장치구와 콤파일러에 의한 명령의 재순서화에 영향을 주므로 조건코드를 처리하는 특별한 론리가 필요 없다. 그대신 관흐름처리장치구가 등록기값의 의존성을 처리할수 있어야 한다. 더우기 등록기파일에 배치된 조건들은 번역시간내에 할당되여 등록기에 기억된 다른 값들로서 재리용한다.

표 12-10. MIPS R-계렬명령모임

Op	설 명	Op	설 명
	넣기 및 기억명령		**곱하기 및 나누기명령**
LB	바이트넣기	MULT	곱하기
LBU	부호 없는 바이트넣기	MULTU	부호 없는 곱하기
LH	반단어넣기	DIV	나누기
LHU	부호 없는 반단어넣기	DIVU	부호 없는 나누기
LW	단어넣기	MFHI	HI 로부터 이동
LWL	왼쪽 단어넣기	MTHI	HI 에로 이동
LWR	오른쪽 단어넣기	MFLO	LO 로부터 이동
SB	바이트기억	MTLO	LO 에로 이동
SH	빈단어기억		**뛰여넘기와 분기명령**
SW	단어기억	J	뛰여넘기
SWL	왼쪽 단어기억	JAL	뛰여넘기와 련결
SWR	오른쪽 단어기억	JR	등록기에로 뛰여넘기
	산수연산명령(ALU 직접값)	JALR	뛰여넘기와 련결등록기
ADDI	직접값더하기	BEQ	같은 경우 분기
ADDIU	부호 없는 직접값더하기	BNE	같지 않은 경우 분기
SLTI	직접값보다 작은 경우에 설정	BLEZ	작거나 령과 같은 경우 분기
SLTIU	부호 없는 직접값보다 작은 경우에 설정	BGTZ	령보다 큰 경우 분기
ANDI	직접값론리곱하기	BLTZ	령보다 작은 경우 분기
ORI	직접값론리더하기	BGTZ	령보다 큰 경우 분기
XORI	직접값안맞음론리더하기	BLTZAL	령보다 작거나 련결인 경우 분기
LUI	웃직접값전개	BGEZAL	령보다 크거나 같고 련결인 경우 분기
	산수연산명령(3-연산수, R-형)		**협동처리장치명령**
ADD	더하기	LWC$_Z$	협동처리장치에 단어넣기
ADDU	부호 없는 더하기	SWC$_Z$	협동처리장치에 단어기억
SUB	덜기	MTC$_Z$	협동처리장치에 이동
SUBL	부호 없는 덜기	MFC$_Z$	협동처리장치에서 이동
SLT	보다 작으면 설정	CTC$_Z$	협동처리장치에 조종을 이동
SLTU	부호 없는 수보다 작으면 설정	CFC$_Z$	협동처리장치에서 조종을 이동
AND	론리곱하기	COP$_Z$	협동처리장치연산
OR	론리더하기	BC$_Z$T	Z 가 참이면 협동처리장치로 분기
XOR	안맞음론리더하기	BC$_Z$F	Z 가 거짓이면 협동처리장치로 분기
NOR	더하기부정		**특수명령**
	밀기명령	SYSCALL	체계호출
SLL	론리왼쪽밀기	BREAK	중지
SRL	론리오른쪽밀기		
SRA	산수오른쪽밀기		
SLLV	론리변수왼쪽밀기		
SRLV	론리변수오른쪽밀기		
SRAV	산수변수오른쪽밀기		

표 12-11. 추가적인 R4000 명령

Op	설 명	Op	설 명
	넣기 및 기억명령		**례외명령**
LL	련결넣기	TGE	보다 크거나 같으면 추적
SC	조건기억	TGEU	부호고려가 없이 보다 크거나 같으면 추적
SYNC	동기	TLT	보다 작으면 추적
	뛰여넘기와 분기명령	TLTU	부호고려가 없이 보다 작으면 추적
BEQL	같으면 분기	TEQ	같으면 추적
BNEL	같지 않으면 분기	TNE	같지 않으면 추적
BLEZL	보다 작거나 령과 같으면 분기	TGEI	직접값보다 크거나 같으면 추적
BGTZL	령보다 크면 분기	TGEIU	부호 없는 직접값보다 크거나 같으면 추적
BGEZL	령보다 작으면 분기	TLTI	직접값보다 작으면 추적
BLTZALL	령보다 작거나 련결이면 분기	TLTIU	부호 없는 직접값보다 작으면 추적
BGEZALL	보다 크고 령과 같거나 련결이면 분기	TEQI	직접값과 같으면 추적
BC_ZTL	z 가 참이면 협동처리장치로 분기	TNEI	직접값과 같지 않으면 추적
CD_ZFL	z 가 거짓이면 협동처리장치로 분기		**협동처리장치명령**
		LDC_Z	2 중협동처리장치넣기
		SDC_Z	2 중협동처리장치기억

대다수 RISC 에 기초한 콤퓨터들과 마찬가지로 MIPS 는 단일한 32bit 명령길이를 가진다. 이 단일한 명령길이는 명령꺼내기와 해신을 쉽게 하며 또 가상기억기관리단위(명령들이 단어 혹은 페지경계를 가로지르지 않는다.)와 명령꺼내기의 호상작용을 간단하게 한다. 세가지 명령형식(그림 12-8)은 명령해신을 간단히 할수 있도록 조작코드와 등록기참조에서 공통적인 형식화를 가진다. 더 복잡한 명령들의 형식은 번역할 때 분석한다.

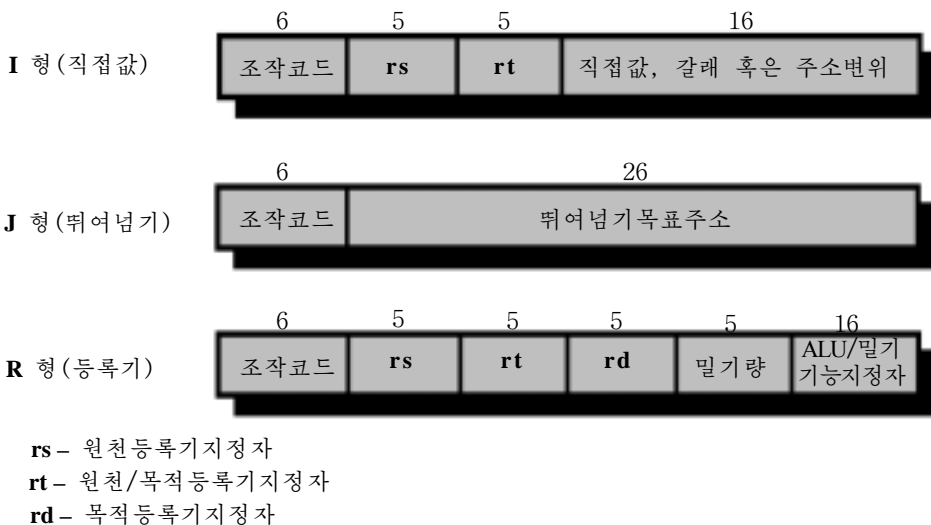



그림 12-8. MIPS 의 명령형식

MIPS 에서는 가장 단순하면서도 가장 빈번히 리용되는 기억기주소지정방식만이 하드웨어로 실현된다. 모든 기억기참조는 32bit 등록기로부터의 16bit 변위로 이루어 진다. 실례로 단어넣기명령은 다음과 같은 양식을 가진다.

lw r2 ,128(r3) : 등록기 3 으로부터 변위된 주소 128 에 있는 단어를 등록기 2 에 넣는다.

모든 32bit 일반등록기들은 기준등록기로 리용할수 있다. 등록기 r0 은 늘 0 을 가진다.

콤파일러는 일반콤퓨터들에서의 전형적인 주소지정방식들을 합성하는 여러 기계명령들을 리용할수 있게 한다. 이와 관련한 몇가지 실례를 표 12-12 에 보여 주었다 [CHOW87]. 이 표는 lui(웃직접값넣기)명령의 리용을 보여 주고 있다. 이 명령은 16bit 직접값을 등록기의 웃절반에 넣고 아래절반은 0 으로 설정한다.

표 12-12. 다른 주소지정방식들을 MIPS 주소지정방식과 합성하기

표면상 명령	실지명령
Lw r2, < 16bit 의 변위 >	lw r2, < 16bit 의 변위 > (r0)
Lw r2, < 32bit 의 변위 >	lui r1, <높은 16bit 의 변위 > lw r2, <낮은 16bit 의 변위 > (r1)
lw r2, < 32bit 의 변위 > (r4)	lui r1, <높은 16bit 의 변위 > addu r1, r1, r4 lw r2, < 낮은 16bit 의 변위 >(r1)

2. 명령관흐름

MIPS 명령구성방식이 단순하므로 MIPS 는 매우 효과적인 관흐름처리를 할수 있다. MIPS 관흐름의 발전과정을 고찰하는것은 일반적인 RISC 관흐름처리의 발전을 설명하는 것이므로 이것을 고찰해 보자.

초기의 실험적인 RISC 체계와 첫 세대의 상품화된 RISC 처리장치들은 체계박자주기마다 하나의 명령을 처리하는 실행속도를 가지고 있다. 이 속도를 개선하기 위하여 두가지 종류의 처리장치들 즉 슈퍼스칼라구성방식과 슈퍼관흐름구성방식에 기초한 처리장치들이 개발되였다. 이 처리장치들은 체계박자주기당 여러개의 명령을 실행할수 있다. 본질적으로 슈퍼스칼라구성방식은 관흐름의 한 단계에서 두개이상의 명령들이 동시에 처리될수 있도록 매 관흐름단계들을 재현시킨다. 슈퍼관흐름구성방식은 보다 조밀하게 세분화된 관흐름단계를 리용할수 있게 하는 구성방식이다. 단계가 많을수록 더 많은 명령들이 같은 시간에 관흐름에 있을수 있으며 이것은 병렬처리화를 증가시키는것으로 된다.

이상의 두가지 방식들은 각기 제한성을 가지고 있다. 우선 슈퍼스칼라관흐름처리에서는 각이한 관흐름들에서 명령들사이 의존성이 체계의 속도를 떨굴수 있다. 또한 간접소비시간론리가 이 의존성을 맞추기 위하여 필요하다. 슈퍼관흐름처리된 처리인 경우에는 한 단계에서 다음단계로 명령들을 전송하는것과 관련한 간접소비시간이 있다.

슈퍼스칼라구성방식에 대한 고찰은 제 13 장에서 진행한다. MIPS R4000 은 RISC 에 기초한 슈퍼관흐름구성방식의 대표적인 실례이다.

그림 12-9 ㄱ는 R3000 의 명령관흐름을 보여 주고 있다. R3000 에서는 관흐름이 박자주기당 한번 전진한다. MIPS 콤파일러는 지연틈의 70~90%를 코드로 채우도록 명령들을 재정돈한다. 모든 명령들은 다음과 같은 5 개의 관흐름단계를 차례로 거친다.

- 명령꺼내기

- 등록기파일로부터 원천연산수꺼내기
- ALU 연산 혹은 자료연산수주소발생
- 자료기억기참조
- 등록기파일에 거꾸로 쓰기

그림 12-9 ㄱ에서 설명하고 있는바와 같이 관흐름처리에는 관흐름처리로 인한 병렬화만이 아니라 단일한 명령실행에서의 병렬화도 있다. 이를 위하여 60ns 의 박자주기를 두개의 30ns 주기단계로 분할한다. 외부명령과 캐쉬에 대한 자료호출연산들은 주요 내부연산들 (OP, DA, IA)과 같이 모두 60ns 를 요구한다. 명령해신은 하나의 30ns 주기단계만을 요구하는 단순한 연산으로서 매 명령에서 등록기꺼내기와 겹쳐 진다. 분기명령인 경우 주소계산에서도 역시 명령해신과 등록기꺼내기가 겹쳐 져 명령 i 에서의 분기가 명령 $i+2$ 의 명령캐쉬(ICACHE)호출을 지정할수 있다. 이와 류사하게 명령 i 에서의 넣기는 명령 $i+1$ 의 연산수로 직접 리용되는 자료를 불러 낸다. 한편 ALU 및 밀기결과는 지연이 없이 명령 $i+1$ 로 직접 넘어 간다. 이와 같이 명령들사이를 긴밀하게 결합시키면 매우 효과적인 관흐름을 만들수 있다.

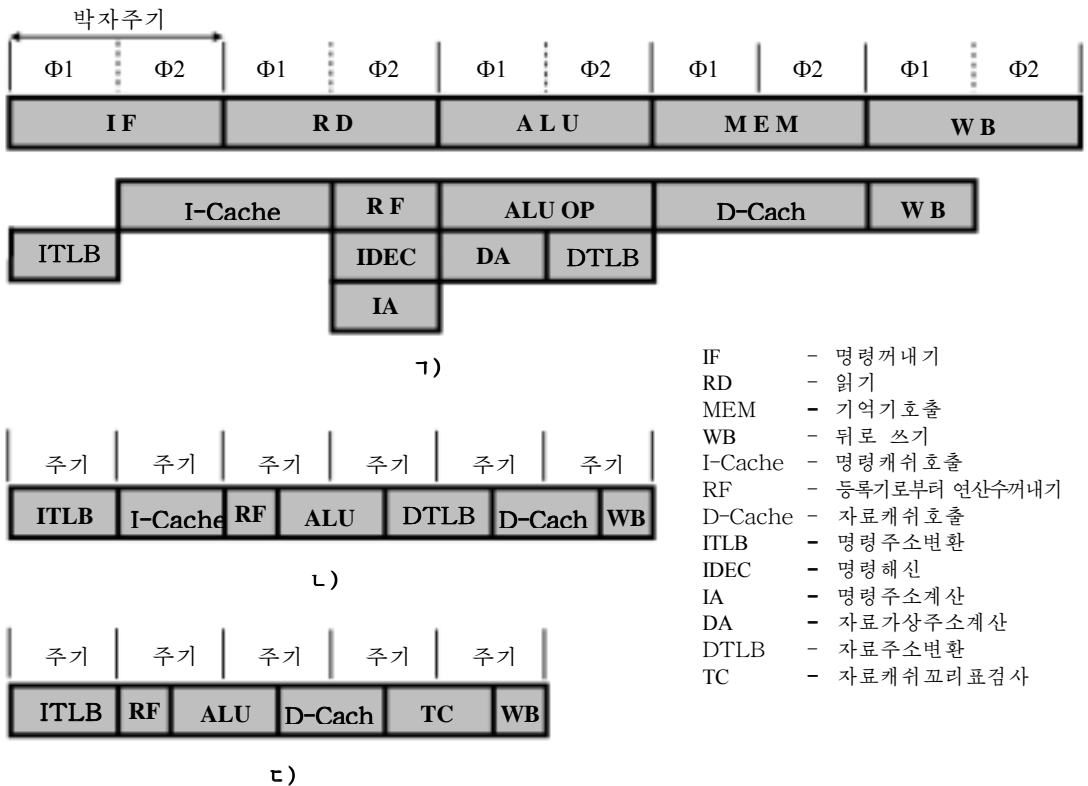



그림 12-9. R 3000 관흐름의 강화
ㄱ-구체화된 R3000 관흐름, ㄴ-단축된 대기시간을 가진 축소 R3000 관흐름,
ㄷ-병렬 TLB 캐쉬호출이 있는 최적 R3000 관흐름

구체적으로는 매 박자주기가 그림에서 Φ1 및 Φ2 로 표시한 개별적인 30ns 주기단계들로 분할된다. 이 매 주기단계에서 수행되는 기능들은 표 12-13 과 같다.

R4000 은 R3000 에 비해 여러가지 기술적인 발전을 가져 왔다. 보다 고급한 기술을

리용하면 체계박자주기시간을 절반으로 즉 30ns 로 줄일수 있으며 등록기파일에 대한 호출시간도 절반으로 단축할수 있다. 이외에도 소편의 더 큰 고밀도화를 실현하면 명령과 자료캐쉬들을 한 소편우에 실현할수 있다. R4000 의 관흐름을 고찰하기전에 마지막으로 R3000 관흐름을 성능이 개선되도록 수정하여 R4000 기술로 어떻게 리용하는가를 보기로 하자.

그림 12-9 ㄴ는 관흐름의 첫 단계를 보여 주고 있다. 우선 이 그림에서 주기는 그림 12-9 ㄱ에서의 주기보다 절반 작다는것을 보여 준다. 명령 및 자료캐쉬가 한 소편내에 있으므로 명령 및 자료캐쉬단계들은 길어서 절반시간(30ns)을 가진다. 총체적으로 이 단계들은 한 박자주기만을 차지한다. 또한 등록기파일의 속도상승으로 등록기읽기와 쓰기 역시 한 박자주기의 절반시간을 차지한다.

표 12-13. R3000 의 관흐름단계

관흐름단계	주기단계	기 능
IF	Φ1	TLB 를 리용하여 명령의 가상주소를 물리주소로 번역한다 (분기판정을 한후)
IF	Φ2	물리주소를 명령주소로 보낸다.
RD	Φ1	명령캐쉬로부터 명령을 복귀 꼬리표들과 불러 낸 명령의 유효성을 비교
RD	Φ2	명령해신, 등록기파일읽기 분기인 경우 분기목표주소계산
ALU	Φ1+Φ2	등록기 대 등록기연산이면 산수 혹은 론리연산을 수행
ALU	Φ1	분기인 경우 그것이 일어 나는가 일어 나지 않는가를 판정,기억기참조(넣기 혹은 기억)인 경우 자료가상주소계산
ALU	Φ2	기억기참조인 경우 자료가상주소를 TLB 를 리용하여 물리주소로 번역
MEM	Φ1	기억기참조인 경우 자료캐쉬에 물리주소보내기
MEM	Φ2	기억기참조인 경우 자료캐쉬로부터 자료를 복귀하고 꼬리표를 검사
WB	Φ1	등록기파일에 쓰기

R4000 캐쉬들이 한 소편우에 있으므로 가상주소 대 물리주소번역은 캐쉬호출을 지연시킬수 있다. 이 지연은 캐쉬를 가상적으로 첨수화된것으로 실현하거나 캐쉬호출과 주소번역을 병렬로 진행하면 감소시킬수 있다. 그림 12-9 ㄷ는 이와 같은 방법을 적용한 최적 R3000 관흐름을 보여 주고 있다. 사건들이 압축되여 있으므로 자료캐쉬꼬리표검사는 캐쉬호출후의 다음주기에서 따로따로 수행된다.

슈퍼관흐름체계에서는 관흐름등록기들을 모든 관흐름단계를 분할하도록 삽입하여 하드웨어로 박자주기당 수배의 리득을 실현하고 있다. 본질적으로 모든 슈퍼관흐름단계는 기본박자주파수의 배수로 동작한다. 이 배수는 슈퍼관흐름처리의 등급에 관계된다. R4000 기술은 등급 2 의 슈퍼관흐름처리를 보장하는 속도와 밀도를 가진다. 그림 12-10 ㄱ는 이 슈퍼관흐름처리를 리용하는 최적화된 R3000 관흐름을 보여 주고 있다. 이 그림은 본질적으로는 그림 12-9 ㄷ와 같은 동적구조이다. 이렇게 하면 성능을 보다 더 개선할수 있다.

R4000 에서는 이 보다 훨씬 더 크고 전용화된 가산기를 설계한다. 이 가산기는 ALU 연산을 2 배의 속도로 실행할수 있다. 이 밖에 넣기 및 기억연산의 속도도 2 배로 높일수

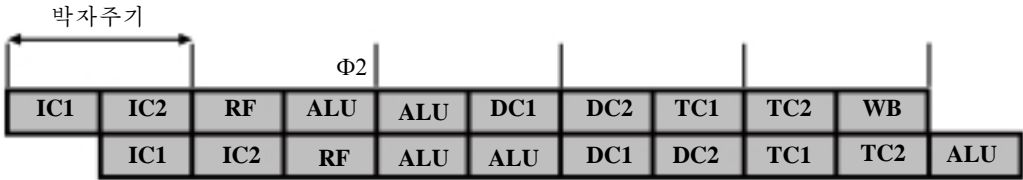
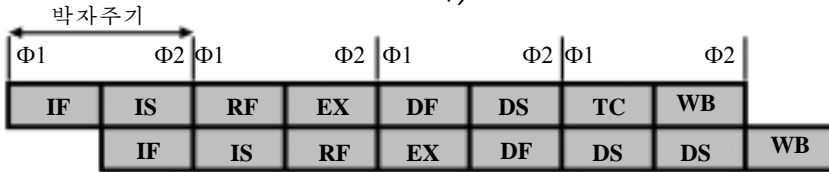
있게 성능을 개선하였다. 이와 같은 결과를 가져오는 관흐름을 그림 12-10 ㄴ에 보여 주었다.

IF = 전반명령꺼내기 IS = 후반명령꺼내기 RF = 등록기로부터 조작코드꺼내기
EX = 명령실행 IC = 명령캐쉬 DC = 자료캐쉬
DF = 전반자료캐쉬 DS = 후반자료캐쉬 TC = 표식자검사

그림 12-10. 리론적인 R3000 과 실지 R4000 의 슈퍼관흐름
ㄱ-최적 R3000 관흐름의 슈퍼관흐름조종실현, ㄴ- R4000 의 관흐름

R4000 은 8 개의 관흐름단계를 가지는에 이것은 8 개의 명령이 같은 시간에 관흐름에 있을수 있다는것을 의미한다. 관흐름은 박자주기당 두 단계의 속도로 전진한다. 이 8 개의 관흐름단계들은 다음과 같다.

- **전반명령꺼내기 :** 가상주소를 명령캐쉬와 변환엿보기완충기(TLB)에 넣는다.
- **후반명령꺼내기 :** 명령캐쉬가 명령을 출구하고 TLB 가 물리주소를 발생한다.
- **등록기파일 :** 세가지 동작이 병렬로 일어 난다.
  - ·명령을 해신하여 련쇄조건들을 검사한다(이 명령은 앞선 명령의 결과에 의존한다.).
  - ·명령캐쉬꼬리부검사가 진행된다.
  - ·연산수들을 등록기파일로부터 불러 낸다.
- **명령실행 :** 세가지 동작중 어느 하나가 일어 날수 있다.
  - ·명령이 등록기 대 등록기연산인 경우 ALU 는 산수 혹은 론리연산을 수행한다.
  - ·명령이 넣기 혹은 기억명령이면 자료가상주소를 계산한다.
  - ·명령이 분기명령이면 분기목표의 가상주소를 계산하고 분기조건을 검사한다.
- **전반자료캐쉬 :** 가상주소를 자료캐쉬와 TLB 에 넣는다.
- **후반자료캐쉬 :** 자료캐쉬가 명령을 출구하고 TLB 가 물리주소를 발생한다.
- **꼬리부검사 :** 넣기 및 기억명령에 대하여 캐쉬꼬리부검사를 수행한다.
- **거꾸로 쓰기 :** 명령의 결과를 등록기파일에 쓴다.

제 7 절. SPARC

SPARC(Scalable Processor ARChitecture)는 Sun Microsystems 에 의해 정의된 구성방식이다. Sun 은 그 자신이 SPARC 를 개발하고 실현시켰을뿐아니라 SPARC 와 호환성이 있는 콤퓨터구성방식들을 다른 콤퓨터장치 및 프로그람관련회사들이 개발하도록 허용하고 있다. SPARC 구성방식은 Berkeley 의 RISC I 기계에 그 뿌리를 두고 있으며 그의 명령모임과 등록기구성은 Berkeley 의 RISC 모형에 기초하고 있다.

1. SPARC 의 등록기묶음

Berkeley 의 RISC 와 마찬가지로 SPARC 는 등록기창문을 리용한다. 매 창문은 24 개의 등록기로 구성되며 창문의 총수는 SPARC 의 실현에 따라 결정되는데 대체로 2~32 개의 범위에 있다.

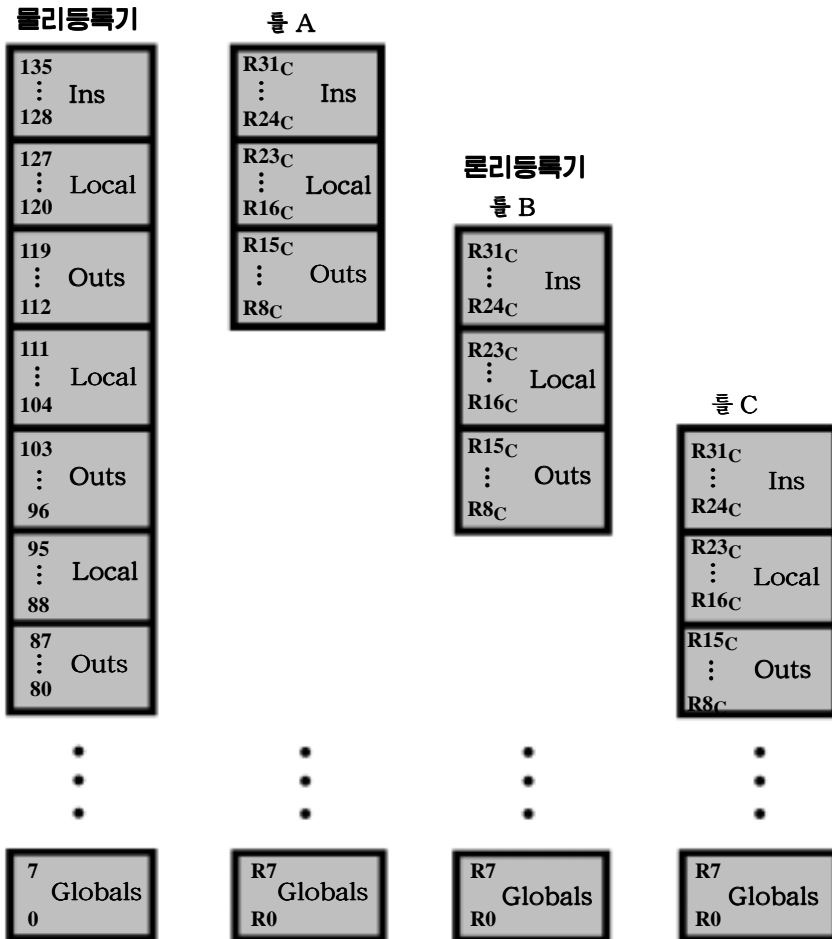



그림 12-11. 3 개의 틀을 가진 SPARC 등록기창문의 지면배치

그림 12-11 은 총 136 개의 물리등록기들을 리용하여 8 개의 창문으로 실현한 등록기구성을 보여 주고 있다. 이미 이 장의 제 1 절에서 고찰한것처럼 이 수는 등록기창문구성에서 합리적인 수로 된다. 물리등록기 0 부터 7 은 모든 틀들에서 공유하는 전역등록기이다. 모든 처리는 론리등록기 0 부터 31 을 참조한다. 그림 12-11 에서 《입구》로 표시된 론리등록기 24 로부터 31 은 호출하는 틀(어미)에서 공유하며 《출구》라는 표식이 붙은 론리등록기 8 부터 15 는 호출된 틀(아들)에서 공유한다. 이 두 부분들은 다른 창문들과 겹치게 된다. 그림에서 《국부》라는 표식이 붙은 론리등록기 16 으로부터 23 은 공동으로 리용되지 않으며 다른 창문들과 겹치지 않는다. 다음으로 이미 제 4 장 제 1 절에서 고찰한바와 같이 파라메터넘기기를 위해 8 개의 등록기를 리용하면 대부분의 경우에 알맞는것으로 된다(실례로 표 12-4 참고).

그림 12-12 는 등록기겹침의 다른 한가지 고찰방법을 보여 주고 있다. 호출하는 틀은 임의의 파라메터들을 그의《출구》등록기들에 넘기도록 배치한다. 그리고 호출된 틀은 이 등록기들을 그의 《입구》등록기들로서 취급한다. 처리장치는 처리장치상태등록기(PSR)에 위치한 현재창문지시기(CWP)를 보존한다. 여기서 처리장치상태등록기는 현재 실행중인 틀의 창문을 가리킨다. 처리장치상태등록기내의 창문무효마스크(WIM)역시 어느 창문이 무효인가를 지적한다.

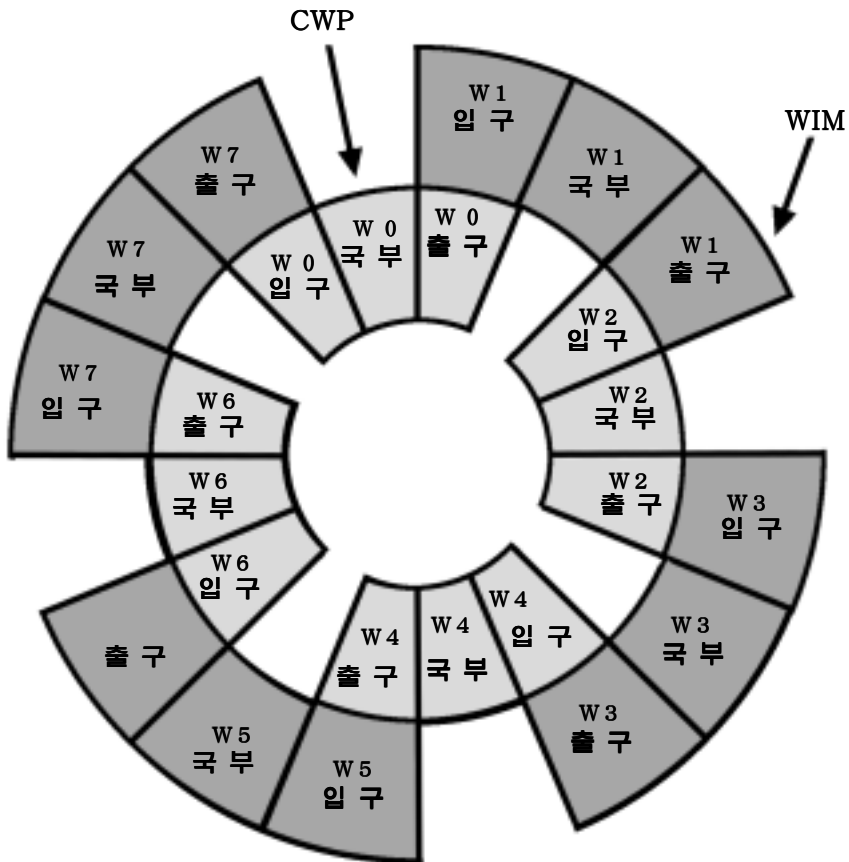



그림 12-12. SPARC 에서 원형탄창을 이루는 8 개의 등록기창문

SPARC 등록기구성방식에서는 자료나 명령을 틀호출을 위해 등록기들에 보관 및 회복시키지 말아야 한다. 그것은 콤파일러가 효과적인 방법으로 틀에 국부등록기들을 할당할뿐 틀들사이 등록기할당에는 무관하게 동작하도록 단순화되였기때문이다.

2. 명령모임

표 12-14 는 SPARC 구성방식에서의 명령들을 보여 주고 있다. 대다수의 명령들은 오직 등록기연산수만을 참조한다. 등록기 대 등록기명령들은 3 개의 연산수를 가지며 다음과 같은 양식으로 표시된다.

$$R_d \leftarrow R_{s1} \ \text{op} \quad S2$$

여기서 R_d 와 R_{s1} 은 등록기참조이며 S2 는 하나의 등록기 혹은 13bit 의 직접값연산수가 될수 있다. 등록기 R_0 은 값 0 으로 고정되여 있다. 우의 이 양식은 국부스칼라변수와 상수들이 대부분 프로그람들에 아주 잘 맞는다.

표 12-14. SPARC 의 명령모임

연산	설명	연산	설명
넣기/기억 명령		**산수연산명령**	
LDSB	부호 있는 바이트넣기	ADD	더하기
LDSH	부호 있는 반단어넣기	ADDCC	더하기 set, icc
LDUB	부호 없는 바이트넣기	ADDX	자리올림이 있는 더하기
LDUH	부호 없는 반단어넣기	ADDXCC	자리올림이 있는 더하기 set,icc
LD	단어넣기	SUB	덜 기
LDD	배단어넣기	SUBCC	덜기 set, icc
STB	바이트기억	SUBX	자리내림이 있는 덜기
STH	반단어기억	SUBXCC	자리내림이 있는 덜기 set, icc
STD	단어기억	MULSCC	다중계단 set, icc
STDD	배단어기억	**뛰여넘기/분기 명령**	
밀기명령		BCC	조건분기
SLL	왼쪽 론리밀기	FBCC	류점수조건분기
SRL	오른쪽 론리밀기	CBCC	협동처리장치조건분기
SRA	오른쪽 산수밀기	CALL	틀호출
론리연산명령		JMPL	뛰여넘기와 련결
AND	론리곱하기	TCC	조건추적
ANDCC	론리곱하기 set,icc	SAVE	등록기창문 전진
ANDN	론리곱하기부정	RESTORE	창문 뒤로 이동
ANDNCC	론리곱하기부정 set,icc	RETT	추적으로부터의 복귀
OR	론리더하기	**기타 명령**	
ORCC	론리더하기 set, icc	SETHI	높은 22 개의 bit 설정
ORN	론리더하기부정	UNIMP	무실행명령(추적)
ORNCC	론리더하기부정 set, icc	RD	특수등록기 읽기
XOR	안맞음론리더하기	WR	특수등록기 쓰기
XORCC	안맞음론리더하기 set, icc	IFLUSH	캐쉬초기화명령
XNOR	안맞음론리더하기부정		
XNORCC	안맞음론리더하기부정 set,icc		

명령모임에서 리용할수 있는 ALU 연산들은 크게 다음과 같이 나누어 볼수 있다.

- 옹근수더하기(자리올림이 있거나 없는것)
- 옹근수덜기(자리내림이 있거나 없는것)
- 비트단위의 론리곱하기(AND), 론리더하기(OR), 안맞음론리더하기(XOR)와 그것들의 부정
- 론리왼쪽밀기, 론리오른쪽밀기 혹은 산수오른쪽밀기

밀기를 제외한 이 모든 명령들은 4 개의 조건코드들(령,부수,자리넘침,자리올림)을 선택적으로 설정할수 있다. 옹근수들은 32bit 로 된 2 의 보수양식으로 표현한다. 또한 단순한 넣기 및 기억명령들만이 기억기를 참조한다.

자료의 형이 단어(32bit), 배단어, 반단어 및 바이트인가에 따라서 서로 다른 넣기 및 기억명령들이 있다. 반단어와 바이트의 경우에는 부호가 있거나 부호가 없는 수들로 따로따로 고찰하여 이 수들을 넣는 명령들이 있다. 부호 있는 수들은 32bit 의 목적등록기를 다 채울 때까지 부호가 확장된다. 부호 없는 수인 경우는 령을 채운다.

등록기주소지정방식외에 리용할수 있는 주소지정방식은 오직 변위주소지정방식뿐이다. 이 방식에서 연산수의 유효주소는 등록기에 있는 주소로부터의 변위와 같다.

$$유효주소 = (R_{s1}) + S2$$

혹은

$$유효주소 = (R_{s1}) + (R_{s2})$$

여기서 두번째 연산수는 직접값 혹은 등록기참조로 된다. 넣기 및 기억을 수행하기 위하여 부가적인 주기단계가 명령주기에 추가된다. 두번째 주기단계에서는 ALU 를 리용하여 기억기주소를 계산한다. 결국 넣기 및 기억은 세번째 주기단계에서 진행된다. 이 주소화방식은 매우 융통성이 있으며 표 12-15 에서 보여 준바와 같이 다른 주소화방식을 합성하는데 리용할수 있다.

표 12-15. SPARC 주소지정방식과 다른 주소지정방식의 합성

방 식	알고리듬	등가적인 SPARC	명 령 형
직접값	연산수=A	S2	등록기-등록기
직적값	유효주소=A	R_0+S2	넣기, 기억
등록기	유효주소=R	R_{S1}, R_{S2}	등록기-등록기
등록기간접	유효주소=(R)	R_{S1} + 0	넣기, 기억
변위	유효주소=(R)+A	R_{S1} + S2	넣기, 기억

SPARC 의 주소지정능력을 MIPS 와 비교해 보면 MIPS 주소화방식은 16bit 의 변위를 리용하지만 SPARC 는 13bit 의 변위를 리용한다. 또한 MIPS 는 두 등록기의 내용으로부터 주소를 얻지 않는다.

3. 명령형식

MIPS R4000 에서와 같이 SPARC 는 32bit 명령형식을 가진 단순한 명령모임을 리용한다(그림 12-13). 모든 명령들은 2bit 의 조작코드로 시작한다. 어떤 명령에서는 그 형식에 추가적으로 조작코드비트들이 확장되기도 한다. 호출명령에서는 2 의 보수양식으로 된 32bit 의 PC 상대주소를 가지도록 30bit 의 직접값연산수의 오른쪽에 두개의 령비트들이

확장된다. 모든 명령들은 이와 같은 양식으로 주소지정을 할수 있도록 32bit 의 한계로 정렬된다.

분기명령은 4개의 표준조건코드비트들에 대응하는 4bit의 조건마당을 포함하며 이것을 리용하여 임의로 결합된 조건들을 검사할수 있다. 22 bit의 PC상대주소는 22 bit의 2의 보수 상대주소를 가지도록 오른쪽에 2개의 령비트가 확장된다. 분기명령의 특수한 특징은 취소비트에 있다. 취소비트가 설정되지 않은 경우에는 분기가 일어 나건 상관없이 늘 분기의 다음명령이 실행된다. 이것은 많은 RISC기계들에서 볼수 있는것으로서 이미 이 장의 제5절의 전형적인 지연분기연산에서 론의한것이다(그림 12-7 참고). 취소비트가 설정된 경우에는 분기명령에 뒤따르는 명령이 분기가 취해 진 조건에서만 실행된다. 처리장치는 명령이 이미 관흐름에 있다고 해도 그 명령의 작용을 억제한다(결과를 취소시킨다.). 이 취소비트는 콤파일러가 조건분기에 뒤따르는 지연틈을 보다 쉽게 메우도록 하는데 효과적으로 리용되는 비트이다. 분기목표가 있는 명령은 분기가

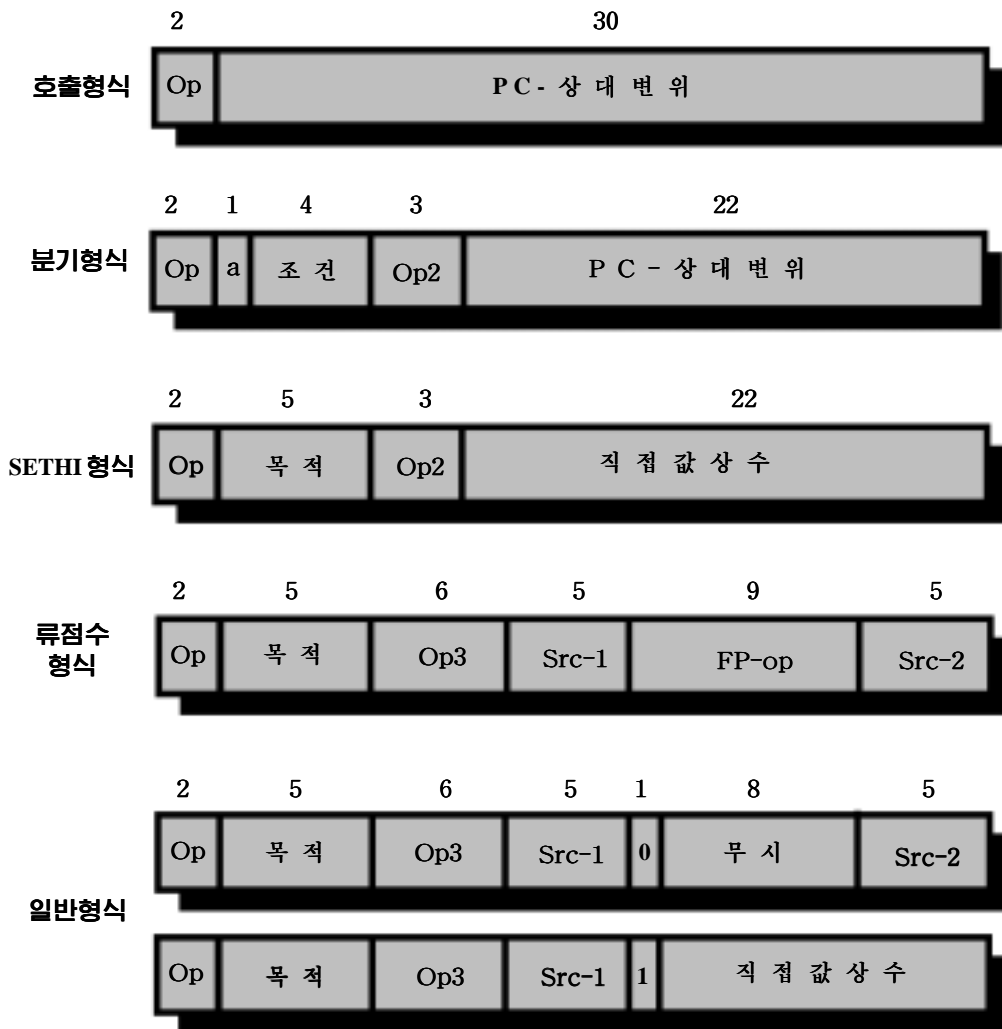



그림 12-13. SPARC 명령형식

일어 나지 않는 경우 그 명령이 취소될수 있으므로 늘 지연틈에 넣어 지게 된다. 이 기술이 필요하게 되는 리유는 조건분기명령들이 일반적으로 그 시간의 절반이상을 차지하기때문이다.

SETHI 명령은 32 bit 의 값을 넣거나 기억시키는데 리용되는 특별한 명령이다. 이 명령은 주소나 큰 상수들을 넣거나 기억하는데 리용된다. SETHI 명령은 22 bit 의 직접값연산수로 등록기의 22 개의 높은자리비트들을 설정하며 낮은 자리 10 개의 비트들에는 0 을 설정한다. 13bit 의 직접값상수는 일반적인 형식들중의 어느 한가지로 규정될수 있으며 이러한 명령은 등록기의 나머지 10 개의 비트들을 메우는데 리용될수 있다. 넣기 혹은 기억명령은 또한 직접주소지정방식을 실현하는데도 리용될수 있다. 기억기의 위치 k 로부터 값을 넣기 위하여 다음과 같은 SPARC 명령들을 리용할수 있다.

```
sethi   %hi(k), %r8          ; 기억기위치 k 주소의 높은자리 22 bit 를 등록기 r8 에 넣기
ld      [%r8+%lo(k)], %r8    ; 위치 k 의 내용을 r8 에 넣기
```

마크로 %hi 와 %lo 는 기억기위치의 적당한 주소비트들을 이루는 직접값연산수들을 정의하는데 리용된다. SETHI 의 리용은 MIPS 의 LDI 명령의 리용과 류사하다(표 12-12).

류점수형식은 류점수연산을 위해 리용된다. 이를 위해 두개의 원천등록기들과 한개의 목적등록기가 지정된다.

넣기, 기억, 산수 및 론리연산과 같은 다른 모든 연산들은 그림 12-13 에서 보여 준 마지막 두가지 형식중 어느 하나를 리용한다. 이 두가지 형식중 하나는 두개의 원천등록기들과 하나의 목적등록기를 리용하며 다른 하나는 하나의 원천등록기, 하나의 13bit 직접값연산수, 하나의 목적등록기들을 리용한다.

제 8 절. RISC 와 CISC 의 비교

오랜 기간에 걸쳐 콤퓨터구성방식과 구성에 대한 일반적인 발전추세는 처리장치의 복잡성을 더 증대시키는 방향이였다. 즉 보다 많은 명령, 보다 많은 주소지정방식, 보다 전용화된 등록기들을 가진 복잡한 처리장치를 만드는 방향으로 발전하였다. RISC 의 성장은 이러한 경향속에서 그의 기본원리를 근본적으로 깨뜨렸다. RISC 체계의 출현과 RISC 의 우점을 극구찬양하는 지지자들에 의해 발표된 론문들은 사실상 콤퓨터구성방식의 기본흐름이라고 할수 있을 정도로 이에 반작용을 일으켰다.

RISC 방식의 우점을 론증한 연구는 크게 두가지 부류로 나누어 진행되였다.

- **정량적인 고찰**:비교기술을 리용하여 RISC 와 CISC 기계들에서 프로그람의 크기와 실행속도비교
- **정성적인 고찰** : 고급언어의 지원과 현존 VLSI 기술의 최적리용 등의 평가

정량적인 평가를 위한 대다수 연구는 RISC 체계에 대한 연구를 진행하는 사람들에 의해 진행되였으며([PATT82b], [HEAT84], [PATT84]) 기본적으로 RISC 방식이 더 좋다는것을 론증하는 방향으로 나갔다. 일부 연구자들은 이 문제를 고찰하여 납득할만한 결과를 주지 못하였다([COLW85a, FLYN87, DAVI87]). 이와 같은 비교를 진행하는데서는 여러가지 문제점들이 있다.

- 생명주기비용, 기술수준, 문회로복잡성, 콤파일러의 부작용, 조작체계의 지원 등을 비교할수 있는 RISC와 CISC 콤퓨터들의 쌍이 없다.
- 프로그람들의 확정적인 검사모임이 존재하지 않는다. 성능은 프로그람에 따라 변한다.
- 콤파일러작성에서의 기량으로 말미암아 효과들중에서 하드웨어효과를 구별하기가 힘들다.
- RISC에 대한 대다수 비교분석은 상품화되여 판매되는 콤퓨터들에서가 아니라 《장난감》기계들에서 진행하였다. 게다가 RISC로써 고찰한 대다수 기계들은 RISC와 CISC의 특성들을 다같이 가지고 있다. 또 RISC의 《장난감》기계들과의 좋은 비교로 될수 있는《순수—놀이》용 CISC 기계 (VAX, Pentium)도 없다.

다음으로 정상적인 평가는 거의 해석과 추측에 의한 평가이다. 여러 연구자들이 이 평가에 관심을 돌렸으나 그 결과는 기껏해서 모호한것이였다. 물론 그 가운데는 일정하게 론거가 있는것도 있고 그렇지 못한것도 있다.

최근년간에 RISC와 CISC의 비교판정은 거의나 하지 않는다. 그것은 기술이 점차적으로 하나로 집초되였기때문이다. 소편밀도가 높아 지고 하드웨어속도가 증가함에 따라 RISC체계도 보다 복잡한 체계로 되였다. 이와 함께 최대성능을 달성하기 위한 노력으로 CISC설계에서도 전통적으로 RISC와 관련된 문제점들에 중점을 두고 있다. 즉 일반등록기수를 늘이며 명령관흐름설계를 강화하는 방향으로 나가고 있다.

참고문헌

RISC에 대한 훌륭한 참고로 될만한 책들은 [WARD90], [PATT98]와[HENN96] 등이다.

[KANE92]는 상품화된 MIPS 기계들을 상세히 고찰하고 있다. [MIRA92]는 MIPS R4000에 대한 전반적인 고찰을 심도 있게 하고 있다. [BASH91]은 R3000 관흐름으로부터 R4000 슈퍼관흐름에 이르기까지의 발전과정을 서술하고 있다. SPARC에 대해서는 [DEWA90]에서 상세히 고찰하고 있다.

BASH91 Bashteen, A.; Lu□.; and Mullan,J." A Superpipeline Approach to the MIPS Architecture." *Proceedings, COMPCON Spring '91,* February 1991.

DEWA90 Dewar, R. , and Smosna, M. *Microprocessors: A Programmer's View.* New York: McGraw-Hill, 1990

HENN96 Hennessy, J., and Patterson, D. *Computer Architecture: A Quantitative Approach.* San Mateo, CA: Morgan Kaufmann, 1996.

KANE92 Kane,G., and Heinrich, J. *MIPS RISC Architecture.* Englewood Cliffs, NJ: Prentice Hall, 1992

MIRA92 Mirapuri, S.; Woodacre,M.; and Vassghi,N. "The MIPS R4000 Processor." *IEEE Micro,* April 1992.

PATT98 Patterson, D., and Hennessy, J. *Computer Organization and Design: The Hardware/Software Interface.* San Mateo,CA: Morgan Kaufmann, 1998.

WARD90 Ward, S.,and Halstead, R. *Computation Structures.* Cambridge, MA: MIT Press, 1990.

련습문제

1. 그림 4-29 의 호출-복귀패턴을 생각하면서 다음의 창문크기에서 얼마만한 자리넘침 혹은 아래자리넘침이 일어 나겠는가를 설명하시오.
   ㄱ) 5?
   ㄴ) 8?
   ㄷ) 16?
2. 그림 12-2 의 론의에서는 창문에서 첫 두개의 부분만이 보관 혹은 회복된다는것을 설명하였다 . 왜 림시등록기들에 보관할 필요가 없는가?
3. 제 12 장 제 5 절에서 고찰한 여러가지 관흐름처리방식을 리용하여 주어 진 프로그람인 경우 그의 실행시간을 결정하려고 한다. 이제

   N =실행명령의 수
   D =기억기호출수
   J =뛰여넘기 명령의 수

   라고 하자. 간단한 순차방식(그림 12-6 ㄱ)인 경우 실행시간은 2N+D 주기단계이다. 두방향, 세방향, 네방향관흐름처리인 경우 그 공식들을 유도하시오.
4. 고급언어에서 다음과 같은 코드쪼각을 고찰해 보자.

```
for I in 1...100 loop
     s ← s+Q(I).VAL
end loop;
```

   이제 Q 가 32byte 의 레코드배렬이고 VAL 마당은 매 레코드의 첫 4 byte 에 있다고 하자. 80486 코드를 리용하면 이 프로그람쪼각을 다음과 같이 번역할수 있다.

```
      MOV    ECX, 1          ; I 를 유지하기 위해 등록기 ECX 를 리용
LP:   IMUL   EAX, ECX, 32    ;EAX 에 변위를 넣기
      MOV    EBX, Q[EAX]     ;VAL 마당을 넣기
      ADD    S, EBX          ;S 에 더하기
      INC    ECX             ;I 의 1 증가
      JNE    LP              ;I=100 까지 순환
```

   이 프로그람은 IMUL 명령을 리용하고 있다. 이 명령은 둘째 연산수를 셋째 연산수인 직접값에 곱하고 그 결과를 첫 연산수에 배치한다(문제 10-13 참고). RISC 지지파는 훌륭한 콤파일러에서는 IMUL 과 같이 불필요하게 복잡한 명령들은 제거할수 있다고 보고 있다. 우의 80486 프로그람을 IMUL 을 리용하지 않고 다시 작성하여 이와 같은 론증을 해보시오.
5. 다음과 같은 순환고리를 고찰해 보자.

```
s: = 0;
for K := 1 to 100 do
        S := S - 1;
```

이것을 일반아쎔블리언어로 직접 번역하면 다음과 같다.

```
      LD     R1, 0              ;S의 값을 R1에 유지
      LD     R2, 1              ;K의 값을 R2에 유지
LP:   SUB    R1, R1, R2         ;S:=S-1
      BEQ    R2, 100, EXIT      ;K=100이면 수행
      ADD    R2, R2, 1          ;기타이면 K를 증가
      JWP    LP                 ;고리의 시작으로 가기
```

RISC 콤퓨터인 경우 콤파일러는 처리장치가 지연분기기구를 채용할수 있도록 이 코드에 지연틈을 도입한다. JMP 명령은 그것이 늘 SUB 명령을 동반하므로 취급하기가 쉽다. 그러므로 JMP명령은 다음의 지연틈에 단순히 SUB명령을 복사해 놓을수 있다. BEQ는 간단치 않다. BEQ 다음에 ADD 명령이 여러번 실행되므로 현재 상태에서는 이 코드를 버릴수 없다. 따라서 NOP 명령이 필요하게 된다. 결과적인 코드를 보여 주시오.

6. 표 12-8에 다음의 처리장치들을 추가적으로 넣으시오.

   ㄱ. Pentium Ⅱ
   ㄴ. PowerPC

7. MIPS 명령모임의 한 부분으로서 목록화되여 있지 않는 일반기계명령들은 많은 경우 단일 MIPS명령으로 합성할수 있다. 다음의 경우에 이것을 보여 주시오.

   ㄱ.등록기 대 등록기이동
   ㄴ.증가,감소
   ㄷ.보수
   ㄹ.부정
   ㅁ.지우기

8. SPARC는 K개의 등록기창문을 가진다. 물리등록기의 수 N이라는것은 무엇인가?

9. SPARC는 CISC콤퓨터들에서 일반적으로 볼수 있는 여러개의 명령들을 가지고 있지 않다. 이들중 일부는 늘 0으로 설정되는 등록기 R0과 산수연산수를 리용하여 쉽게 모방된다. 이렇게 모방된 명령들은 모조명령이라고 하며 이것은 SPARC콤파일러가 인식할수 있는 명령이다. 다음의 모조명령들이 단일한 SPARC명령으로 어떻게 모방되였는가를 보여 주시오.
   여기서 src와 dst는 등록기들을 의미한다(암시:R0에 대한 기억은 무의미하다.).

ㄱ. MOV src, dst	ㄹ. NOT dst	ㅅ. DEC dst
ㄴ.COMPARE src1,src2	ㅁ. NEG dst	ㅇ. CLR dst
ㄷ. TEST src1	ㅂ. INC dst	ㅈ. NOP

10. 다음의 코드쪼각을 고찰해 보자.

```
if k>10
   L:=k+1
else
```

L:=k-1;

SPARC 아셈블러로 이 명령문을 간단히 번역하면 다음과 같은 양식으로 된다.

```
        sethi   %hi(k),%r8          ; 기억기위치 k 의 주소의 높은자리 22bit 를 등록
                                      기 r8 에 넣기
        ld      [%r8+%l0(k)],%r8    ; 위치 k 의 내용을 r8 에 넣기
        cmp     %r8,10              ; r8 의 내용을 10 과 비교
        ble     L1                  ; (r8)≦10 이면 분기
        nop
        sethi   %hi(k),%r9
        ld      [%r9+%l0(k)],r9     ; 위치 k 의 내용을 r9 에 넣기
        inc     %r9                 ; (r9)에 1 을 더하기
        sethi   %hi(L),%r10
        st %r9, [%r10+%l0(L)        ; (r9)를 위치 L 에 기억
        b  L2
        nop
  L1:   sethi   %hi(k),%r11
        ld      [%r11+%l0(k)],%r12  ; 위치 k 의 내용을 r12 에 넣기
        dec     %r12                ; (r12)에서 1 을 덜기
        sethi   %fi(L),%r13
     st  %r12,[%r13+%l0(L)];  (r12)을 위치 L 에 기억
  L2:
```

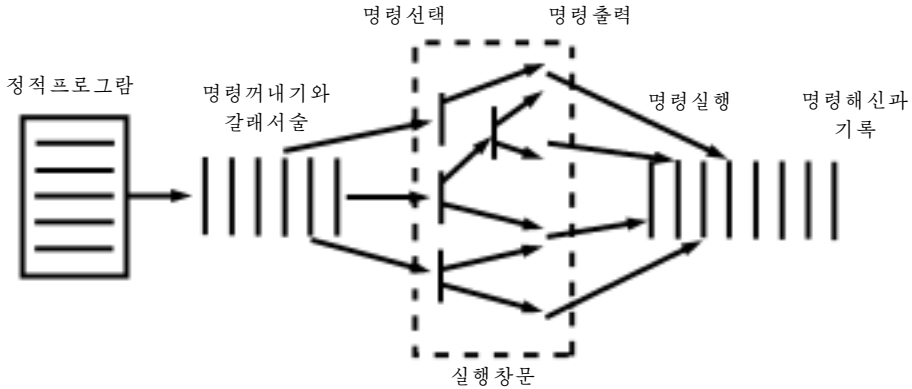
이 코드는 지연분기연산을 허가하는 매 분기명령의 다음에 nop 명령을 가지고 있다.

ㄱ. RISC 콤퓨터와는 아무러한 관계도 없는 표준콤파일러최적화는 일반적으로 우의 코드에 대하여 두가지 변환을 수행할수 있는것으로 하여 효과적이다. 우의 코드에서 두개의 넣기명령은 불필요한것이며 두개의 기억명령은 코드내의 각이한 위치로 기억을 옮기는 경우에는 빠질수 있다. 이 두 변화를 진행한후의 프로그람을 보여 주시오.

ㄴ. 우의 코드에서는 SPARC 에 고유한 일부 최적화를 진행할수 있다. ble 후의 nop 는 지연틈안으로 다른 명령을 옮기거나 ble 명령에 대한 취소비트를 설정하여 재배치할수 있다. 이와 같은 변화를 한후의 프로그람을 보여 주시오.

ㄷ. 우의 코드에는 불필요한 명령들이 있다. 이 불필요한 명령들을 제거한 결과적인 프로그람을 보여 주시오.

제 13 장. 명령준위병렬화와 슈퍼스칼라처리장치

◆◆

◆ 슈퍼스칼라처리장치는 여러개의 독립적인 관흐름을 리용한 처리장치이다. 매개 관흐름은 여러개의 단으로 구성되여 있으므로 한번에 여러개의 명령을 조종할수 있다. 다중관흐름은 새로운 병렬화를 도입하여 여러개의 명령을 한번에 처리할수 있게 하고 있다.

◆ 슈퍼스칼라처리장치는 명령준위병렬화를 리용하였는데 이것은 프로그람의 명령들을 병렬로 집행하자는데 있다. 슈퍼스칼라처리장치는 일반적으로 한번에 여러개의 명령을 읽어 들이는데 이때 독립적이면서도 병렬로 집행할수 있는 린접명령들을 탐색한다. 어느 한 명령에 대한 입구가 선행한 명령의 출력에 의존하는 경우 그다음명령은 같은 시간에 혹은 선행명령전에 실행될수 없다. 일단 그러한 독립성이 식별되면 처리장치는 초기의 기계코드와는 다른 순서로 명령을 출력하고 실현할수 있다.

◆ 처리장치는 보충된 등록기들을 리용하든가 원천코드안에서 등록기참조의 이름을 바꾸는 방법으로 일부 필요 없는 자료의존성을 제거할수 있다.

◆ 순수한 RISC 처리장치는 명령관흐름을 최대로 리용하기 위하여 흔히 지연분기명령을 리용하는데 이 방법은 슈퍼스칼라기계에는 적합하지 않다. 대신 대부분의 슈퍼스칼라기계들은 효과성을 개선하기 위하여 전통적인 분기기술을 리용한다.

◆◆

처리장치기본방식의 하나인 슈퍼스칼라방식이 실현됨으로써 공통명령들인 옹근수와 류점수연산, 넣기, 기억 그리고 조건분기명령들을 동시에 초기화하여 독립적으로 실행할수 있다. 이 방식의 설계에서는 명령관흐름과 관련된 일련의 복잡한 문제가 제기된다.

슈퍼스칼라설계는 RISC 기본방식에 거의 접근하고 있다. RISC 기계의 단순한 명령모임방식에 슈퍼스칼라기술을 받아 들이면 슈퍼스칼라방법을 CISC 기본방식에서도 리용할수 있다.

IBM801 과 Berkeley RISC 와 같이 실지 RISC 연구를 시작한 때로부터 상품화된 RISC 기계가 나올 때까지는 7～8 년이라는 기간이 걸렸지만 첫 슈퍼스칼라기계가 상품화된것은 슈퍼스칼라라는 말이 세상에 나온 때로부터 1～2 년사이이다. 슈퍼스칼라방법은 오늘날 고

성능극소형처리장치를 실현하는데서 표준적인 방법으로 되고 있다.

이 장에서는 먼저 고속관흐름기술과 대조를 이루는 슈퍼스칼라방법을 개괄적으로 설명한 다음 슈퍼스칼라실현과 관계되는 기본설계항목을 보여 준다. 그다음 슈퍼스칼라기본방식의 여러가지 주요실례들을 고찰한다. 마지막으로 슈퍼스칼라설계의 강화된 기술을 보여 주는 새로운 IA-64 기본방식을 고찰한다.

제 1 절. 일반개념

슈퍼스칼라라는 말이 세상에 처음 나온것은 1987[AGER87]년이였는데 이것은 스칼라명령의 실행성능을 개선하기 위하여 설계된 기계를 의미하는것이였다. 이 용어는 제 16 장에서 론의하게 되는 벡토르처리장치들과는 대조를 이룬다. 거의 모든 응용프로그람들에서 연산의 대부분은 스칼라량이다. 따라서 슈퍼스칼라방법은 고성능일반목적처리장치들의 발전단계에서 다음단계라고 말할수 있다.

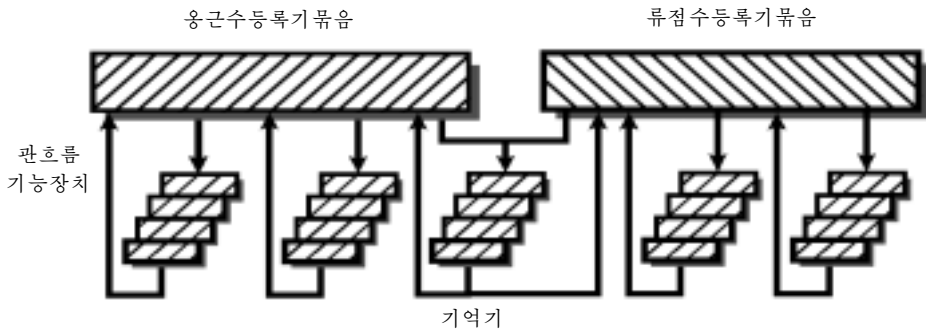



그림 13-1. 일반적인 슈퍼스칼라구성[COME95]

표 13-1. 고속스칼라결합 콤퓨터들의 속도증가

참조	속도증가
[TJAD70]	1.8
[KUCK72]	8
[WEIS84]	1.58
[ACOS86]	2.7
[SOHI90]	1.8
[SMIT89]	2.3
[JOUP89b]	2.2
[LEE91]	7

슈퍼스칼라방법에서 기본요점은 서로 다른 관흐름에서 명령을 독립적으로 실행할수 있는 능력이다. 이 원리는 명령들이 프로그람순서와는 차이나는 순서로 집행되게 한다. 그림 13-1 은 일반적인 슈퍼스칼라구성을 보여 주고 있다. 그림에는 여러개의 기능장치들이 있는데 그것들의 개개는 관흐름으로 되여있으며 여러개의 명령에 대한 병렬실행을 지원한다. 이 실례에서는 두개의 옹근수, 두개의 류점수 그리고 하나의 기억기조작(읽기 혹은 쓰기)들을 같은 시간에 집행하고 있다.

많은 연구자들이 슈퍼스칼라처리장치들을 발명하였는데 그들의 연구결과는 일정한 정도로 성능을 개선할수 있다는것을 보여 주고 있다. 표 13-1 은 이러한 성능들의 우점을 보여 주고 있다. 성능결과의 차이는 모의하는 콤퓨터의 하드웨어와 응용프로그람의 차이로부터 생긴것이다.

1. 슈퍼스칼라와 슈퍼관흐름

보다 높은 성능을 실현하는 다른 한가지 방법은 1988[JOVP88] 년에 처음으로 나온 용어인 슈퍼관흐름이라고 말할수 있다. 슈퍼관흐름처리는 많은 관흐름단들을 리용하면 요구하는 과제들을 박자주기의 절반보다 적은 시간으로 수행할수 있다는데 기초하고 있다. 따라서 2 배의 내부박자주기속도를 가지고 하나의 외부박자주기내에 두개의 과제를 수행하는것으로 된다. 이 방법에 대한 하나의 실례로 MIPS R4000 을 들수 있다.

그림 13-2 에서는 이 두 방법을 비교하였다. 그림의 웃부분은 비교기준으로 리용한다. 비교의 기준으로 되는 관흐름은 한박자주기에 하나의 명령을 출력하므로 한박자주기에

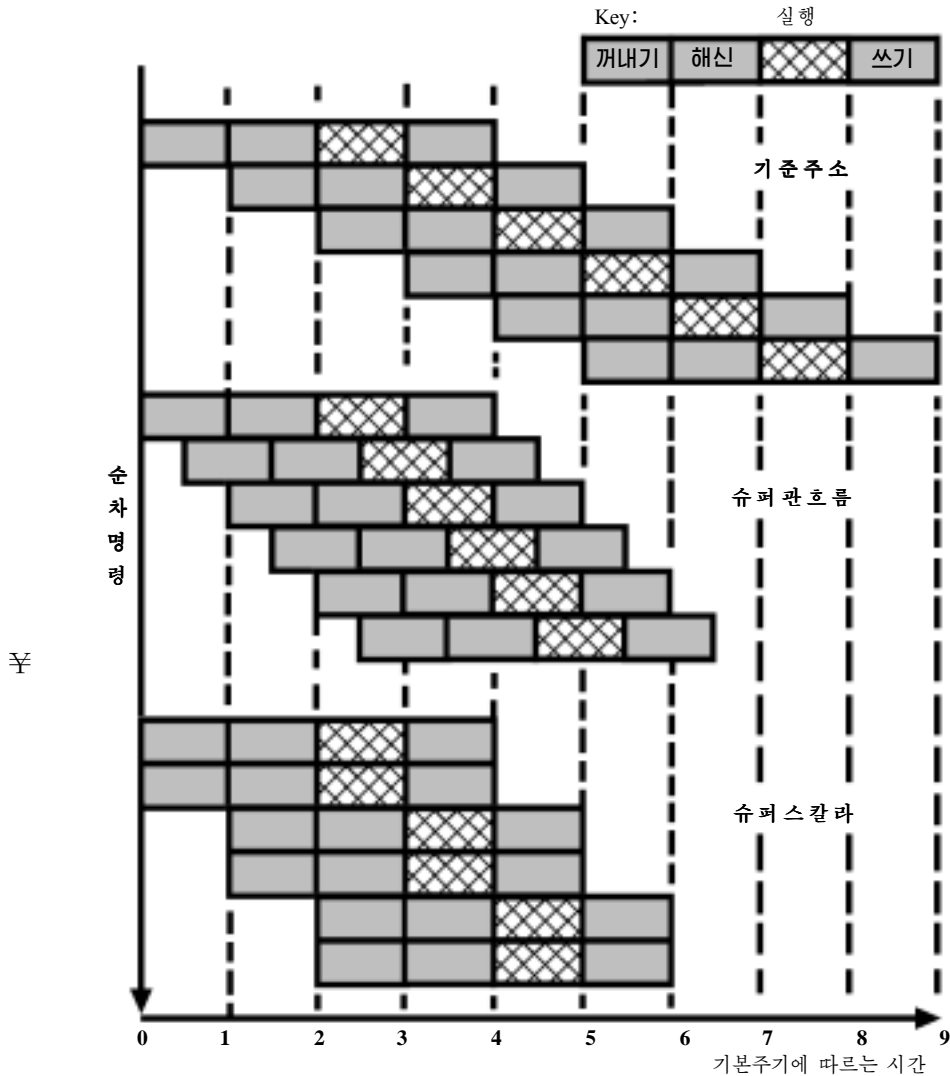



그림 13-2. 슈퍼스칼라와 슈퍼관흐름의 비교

하나의 관흐름단을 수행할수 있다. 관흐름은 명령꺼내기, 명령해신, 명령실행, 결과쓰기의 4 개 단으로 되여 있다. 그림에서 실행단은 그물모양으로 표시하였다. 고려할것은 여러개

의 명령을 동시에 실행하고 있다고 해도 어느 한순간에 실행단에는 하나의 명령만이 있을수 있다는것이다.

그림의 다음부분은 한박자주기에 두개의 관흐름을 수행할수 있는 슈퍼스칼라실현을 보여 주고 있다. 여기에서 주목하여야 할 문제는 매개 단에서 수행되는 기능들은 두개의 겹치지 않는 부분들로 나누어 지는데 그 개개는 반박자주기내에 실현할수 있다는것이다. 이 류형에서 취하게 되는 슈퍼관흐름실현은 2 부류라고 말할수 있다. 마지막으로 그림의 제일 아래부분은 매개 단의 두 요구를 병렬로 실행할수 있는 슈퍼스칼라실현방식을 보여 주고 있다. 물론 보다 높은 등급의 슈퍼관흐름과 슈퍼스칼라도 실현할수 있다.

그림 13-2 에서 보여 준 슈퍼관흐름과 슈퍼스칼라실현방식에서는 안정상태에서 같은 시간에 수행할수 있는 명령수는 같다. 슈퍼관흐름처리된 처리장치는 프로그람의 시작과 모든 분기실행에서 슈퍼스칼라처리장치보다 속도가 떨어 진다.

2. 제한성

슈퍼스칼라방식은 여러개의 명령을 병렬로 실행할수 있는가 없는가에 관계된다. **명령준위병렬화**라는 말은 일반적으로 프로그람을 이루는 명령들이 병렬로 집행될수 있는 정도를 의미하는것이다. 콤파일러에 기초한 최적화와 하드웨어기술의 조합에 의하여 명령준위의 병렬화를 높은 수준에서 실현할수 있다. 령준위의 병렬화를 높이기 위하여 슈퍼스칼라기계에 리용되는 설계기술을 시험하기에 앞서 체계가 대처하여야 할 병렬화에 미치는 기본제한성들을 잘 고찰하여야 한다. 참고문헌 [JOHN91]에는 다음과 같은 5 개의 제한성을 제기하였다.

- 실자료의존성
- 처리절차의존성
- 자원충돌
- 출력의존성
- 반의존성

아래로 내려 가면서 이 제한성가운데서 앞의 세가지를 먼저 론의한다. 마지막두가지에 대한 론의는 다음절에서 한다.

실자료의 의존성

다음과정을 고찰해 보자.

```
add    r1, r2   ; 등록기 r2 의 내용을 등록기 r1 의 내용과 더하여 등록기 r1 에
                  넣기
move   r3, r1   ; 등록기 r1 의 내용을 등록기 r3 에 넣기
```

두번째 명령은 읽기와 해신은 되지만 첫번째 명령이 실행될 때까지 실행할수 없다. 그 리유는 두번째 명령이 첫번째 명령에 의하여 만들어 지는 자료를 요구하기때문이다. 이것을 실자료의존성이라고 한다(흐름의존성 혹은 쓰기－읽기의존성이라고도 한다.).

그림 13-3 에는 2 부류 슈퍼스칼라기계에서의 의존성을 보여 주었다. 의존성이 없으면 두 명령은 읽기와 실행을 병렬로 할수 있다. 만일 첫번째와 두번째명령사이에 자료의존성이 있으면 두번째 명령은 첫번째 명령의 실행이 끝날 때까지 걸리는 박자주기만큼 지연된다. 일반적으로 임의의 명령은 입력값이 모두 처리될 때까지 지연된다.

단순한 스칼라관흐름에서는 앞에서 서술한 명령실행과정에 지연이 생기지 않는다. 이

제 등록기로부터의 읽기가 아니라 기억기로부터의 읽기인 경우를 보기로 하자.

```
Load  r1,  eff   ; 유효기억주소 eff의 내용을 등록기 r1에 넣기
Move  r3,  r1    ; r1의 내용을 r3에 넣기
```

전형적인 RISC처리장치는 기억기로부터의 읽기를 수행하는데 둘이상의 주기가 요구되는데 그것은 소편(처리장치)밖의 기억기나 캐쉬를 호출할 때의 지연시간이다. 이 지연을 보상하는 한가지 방법은 콤파일러가 기억기읽기에 의존하지 않는 하나 혹은 그이상의 련속명령들이 관흐름을 통하여 흐를수 있도록 명령을 재순서화하는것이다. 이 방법은 슈퍼스칼라관흐름인 경우에는 효과가 적다. 넣기하는 동안 진행되는 독립적인 명령들은 거의 첫번째 넣기주기때에 집행되며 넣기가 끝날 때까지 아무것도 하지 않는다.

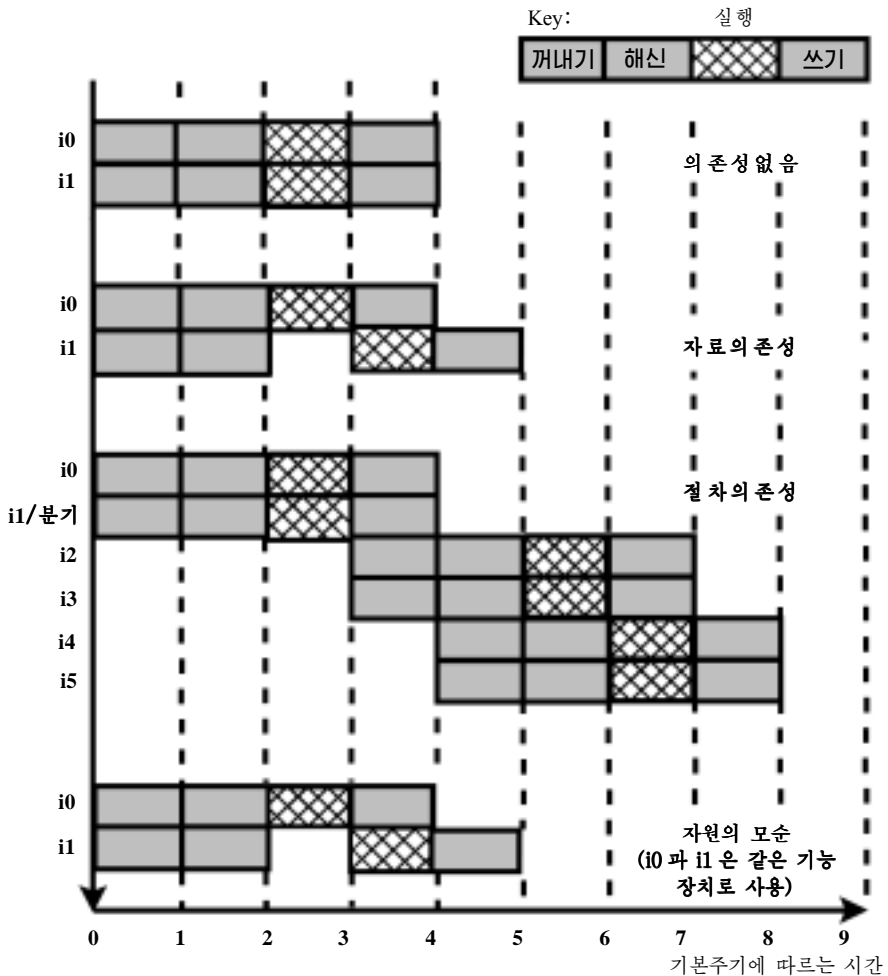



그림 13-3. 의존성의 효과

처리절차의 의존성

제 11 장에서 언급된것처럼 명령실행과정에 분기명령이 나타나면 관흐름조작이 복잡해진다. 분기명령에 뒤따르는 명령들은 분기조작에 대하여 처리절차의 의존성을 가지며 분

기조작이 실행될 때까지 실행될수 없다. 그림 13-3 은 2 부류 슈퍼스칼라관흐름에 대한 분기조작의 효과성을 보여 준다.

이러한 류형의 처리절차의존성은 스칼라관흐름에 영향을 준다. 슈퍼스칼라관흐름의 결과에는 보다 더 큰 영향을 주는데 그것은 적지 않은 시간을 지연으로 하여 잃어 버리기때문이다.

가변길이명령을 리용하면 다른 류형의 처리절차의존성이 생긴다. 왜냐하면 임의의 부분명령의 길이를 모르기때문에 다음명령을 읽기전에 최소한 부분적으로 해신하지 않으면 안되기때문이다. 이것은 슈퍼스칼라관흐름에서 요구되는 동시적인 읽기에 나쁜 영향을 준다. 이것이 슈퍼스칼라기술을 고정된 명령길이를 가진 RISC 또는 RISC 류형의 기본방식에 리용하는 하나의 리유이다.

자원충돌

자원충돌이란 같은 시간에 같은 자료에 대하여 둘 혹은 그이상의 명령들이 경쟁하는 것을 말한다. 자원에는 기억기, 캐쉬, 모선, 등록기－파일포구와 기능장치(례하면 산수-론리연산장치의 더하기회로) 들이 속한다.

관흐름의 견지에서 볼 때 자원충돌은 자료의존성동작과 류사하다(그림 13-3). 그러나 여기에는 몇가지 차이점이 있다. 한가지 차이점은 실자료의존성을 제거할수는 없지만 자원충돌은 자원중첩의 방법으로 극복할수 있다는것이다. 다른 하나의 차이점은 연산이 오랜 시간 진행되는 경우 자원충돌을 적당한 기능장치들을 관흐름처리하여 최소화할수 있다는것이다.

제 2 절. 설계의 문제

1. 명령준위병렬화와 기계병렬화

[JOUP89a]은 명령준위병렬화와 기계준위병렬화의 두 련관개념들사이의 중요한 차이점을 제기하였다. **명령준위병렬화**는 순차적인 명령들이 서로 독립이고 따라서 겹침에 의하여 병렬로 실행될수 있을 때 가능하다.

명령준위병렬화의 개념을 리해하기 위하여 다음의 두 코드쪼각을 고찰해 보자[JOUP89b]:

```
Load   R1  ←  R2            ADD    R3   ←  R3, "1"
ADD    R   ←  R3, "1"       ADD    R4   ←  R3, R2
ADD    R4  ←  R4, R2        Store  [R4] ←  R0
```

왼쪽의 세 명령은 독립이며 리론적으로는 셋이 다 병렬로 실행될수 있다. 반대로 오른쪽 세 명령은 병렬로 실행될수 없다. 왜냐하면 두번째 명령은 첫번째 명령의 결과를 리용하며 세번째 명령은 두번째 명령의 결과를 리용하기때문이다.

명령준위병렬화는 코드안에 실자료의존성과 처리절차의존성이 자주 나타나는가에 따라 결정된다. 이 요소들은 명령모임기본방식과 응용프로그람에 의존한다. 명령준위병렬화는 또한 조작도달시간에 의하여 결정된다. 조작도달시간은 명령의 결과가 다음명령의 연산수로 리용될수 있을 때까지의 시간이다. 조작도달시간은 자료 혹은 처리절차 의존성이 얼마만한 지연을 일으키는가에 따라 결정된다.

기계준위병렬화는 명령준위의 병렬화를 리용하는 처리장치의 능력을 규정하는 척도이

다. 기계병렬화는 같은 시간에 읽거나 실행할수 있는 명령의 수(병렬관흐름의 수)와 독립인 명령들을 찾는데 리용하는 기구와 수법에 의하여 결정된다.

명령준위병렬화와 기계병렬화는 둘다 성능을 높이는데서 중요한 요인이다. 프로그람은 기계병렬화를 원만히 리용하기 위한 명령준위병렬화를 충분히 가지고 있지 못할수 있다. RISC 와 같이 고정길이명령방식을 리용하면 명령준위병렬화를 강화할수 있다. 다른 한편 제한된 기계병렬화는 프로그람성질이 어떠하든 성능을 제한한다.

2. 명령초기화규약

이미 언급한것처럼 기계병렬화는 매개 관흐름단계에서 제기되는 여러개의 사건들을 거의 가지고 있지 않다. 처리장치는 명령준위병렬화를 식별할수 있어야 하며 명령의 읽기, 해신, 실행을 병렬로 조종할수 있어야 한다.

명령실행초기화는 처리장치의 기능장치들안에서 명령실행을 초기화하는 과정을 말하며 **명령초기화규약**은 명령실행을 초기화하는 규약을 의미한다.

중요한것은 처리장치가 관흐름에 들어 와 집행될수 있는 명령이 놓여 있는 현실행점을 찾는것이다. 이 고찰에서 세가지류형의 순서가 중요하게 고려되고 있다.

- 명령을 꺼내는 순서
- 명령을 실행하는 순서
- 명령이 등록기와 기억주소의 내용을 갱신하는 순서

처리장치가 세련되면 될수록 이 순서들사이의 관계는 더욱더 유연해 진다. 여러가지 관흐름요소들의 리용을 최적화하기 위하여 처리장치는 엄격한 순차실행에서 찾아 보게 되는 순서짜기의 순서들가운데서 하나 혹은 몇개를 바꾸어야 한다. 처리장치에 대한 하나의 요구는 그 결과가 정확하여야 한다는것이다. 따라서 처리장치는 앞에서 언급된 여러가지 의존성과 충돌에 적응되여야 한다.

일반적으로 슈퍼스칼라명령실행초기화규약은 다음과 같이 분류하여 묶을수 있다.

- 순차실행의 순차완료
- 순차실행의 비순차완료
- 비순차실행의 비순차완료

순차실행의 순차완료

제일 간단한 명령실행규약은 결과가 얻어 질수 있는 정확한 순서로 명령을 순차실행하며 그와 같은 순서로 결과를 쓰기하는것이다. 그러나 스칼라관흐름에서는 이러한 규약을 쓰지 않는다. 이 규약은 보다 개선된 방식들을 비교하기 위한 기준으로 리용한다.

그림 13-4 ㄱ에 이 규약에 대한 실례를 주었다. 여기서 알수 있는것은 슈퍼스칼라관흐름은 3 개의 개별적인 기능장치들(례하면 옹근수연산, 류수연산)과 재쓰기관흐름단의 두 요구에 의하여 한번에 두개의 명령에 대한 읽기와 해신을 동시에 할수 있다는것이다. 실례에서는 6 개의 명령코드부분에 대하여 다음과 같이 분류하였다.

- 11 은 그 실행에 두 주기를 요구한다.
- 13 과 14 는 같은 기능장치를 요구하므로 충돌한다.
- 15 는 14 에 의하여 처리되는 값에 의존한다.

• 15와 16은 한개의 기능장치를 요구하므로 충돌한다.

한번에 두개의 명령이 나와 해신장치에 들어 간다. 명령들은 쌍으로 읽어 지므로 그 다음 두개 명령은 한쌍의 해신관흐름단이 지워 질 때까지 대기하여야 한다. 기능장치에 대한 충돌이 있거나 기능장치들이 결과를 출력하는데 한주기이상을 요구하는 경우에 순차완료를 보장하기 위하여 명령실행을 잠시 중지한다.

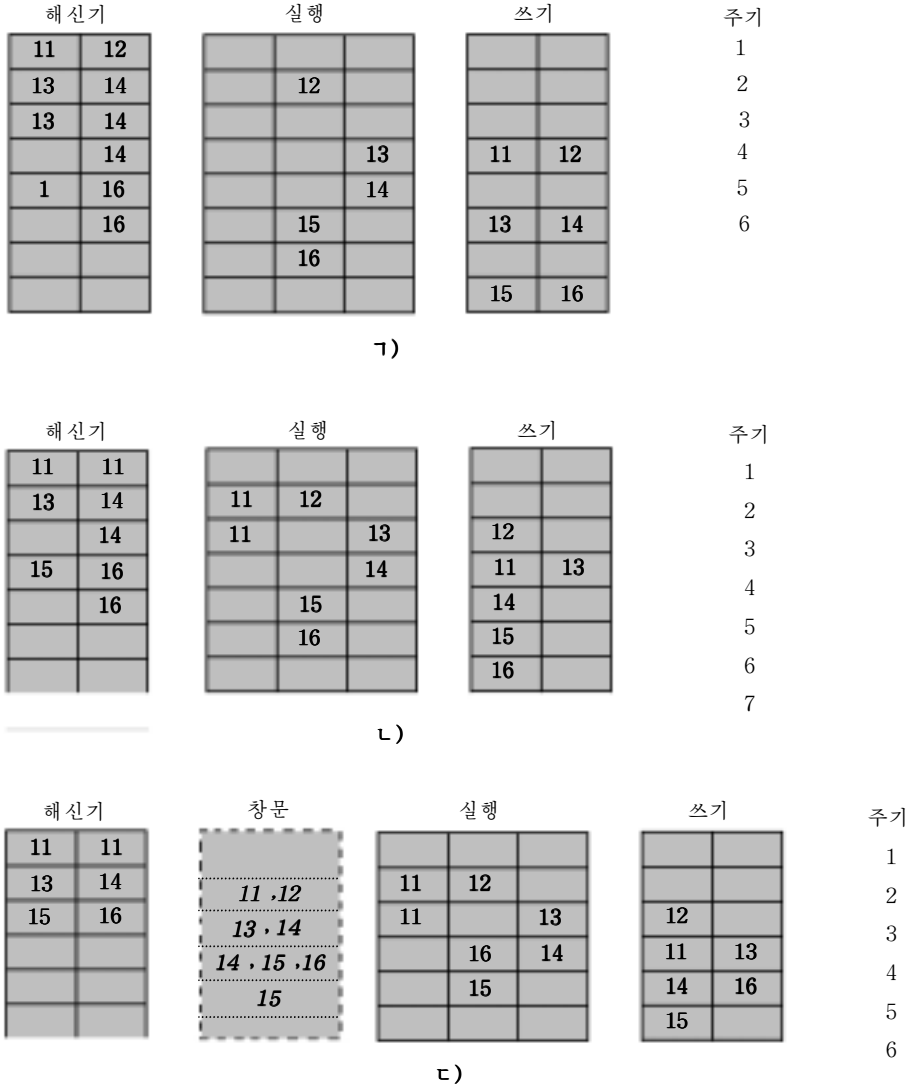



그림 13-4. 슈퍼스칼라명령의 관흐름출력와 완료방식
ㄱ-순차관흐름출력와 순차완료, ㄴ-순차관흐름출력와 비순차완료,
ㄷ-비순차관흐름출력와 비순차완료

이 실례에서 첫번째 명령의 해신으로부터 마지막결과를 쓰기할 때까지 8 박자주기를 가진다.

순차실행의 비순차완료

비순차완료는 스칼라 RISC 처리장치들에서 리용되는데 그것은 여러개의 박자주기를 요구하는 명령의 성능을 개선한다. 그림 13-4 ㄴ는 슈퍼스칼라처리장치에서 비순차완료의 리용을 보여 준다. 명령 12 는 명령 11 에 앞서서 실행이 완료되여야 한다. 그러자면 한주기를 절약하기 위하여 13 이 먼저 완료되여야 한다.

비순차완료를 가지므로 임의의 수의 명령들은 모든 기능장치들로부터 최대급의 기계병렬화에 이르기까지 임의의 시각에는 실행단안에 있게 된다.

명령순차실행은 자원충돌, 자료의존성이든가 처리절차의존성에 의하여 지연된다.

앞에서 언급한 제한성들과 함께 **출력의존성**이라고 하는 새로운 의존성이 제기되고 있다(이것을 일명 **쓰기－쓰기의존성**이라고 부른다.).

다음의 코드부분들은 이 의존성을 보여 주고 있다(OP 는 연산을 의미한다).

```
11 : R3 ← R3 OP R5
12 : R4 ← R3  +  1
13 : R3 ← R5  +  1
14 : R4 ← R3 OP R4
```

명령 12 는 명령 11 보다 먼저 실행할수 없는데 그것은 11 에서 처리된 결과가 등록기 R3 에 넣어 져야 하기때문이다. 이것이 제 1 절에서 서술된 실지자료의존성의 실례이다. 마찬가지로 14 는 13 이 실행될 때까지 기다려야 한다. 그것은 13 에 의하여 만들어 진 결과를 리용하여야 하기때문이다. 11 과 13 사이에는 어떤 관계가 있는가? 그것은 사용자가 정의했기때문에 여기에 자료의존성은 없다. 그러나 13 이 11 보다 앞서서 실행완료를 하였다면 R3 안의 틀린 값이 14 의 실행단계에서 읽기된다. 계속하여 13 은 정확한 출력값을 얻은 다음에 13 이 완료되여야 한다. 이것을 확보하기 위하여 세번째 명령의 출력은 그 결과가 실행시간이 길어 진 선행명령에 의하여 후에 겹쳐쓰기되는 경우 어느 정도 지연되여야 한다.

비순차완료는 순차완료보다 더 복잡한 명령실행론리를 요구한다. 더우기 새치기와 례외명령을 취급하는것은 더 어렵다. 새치기가 발생하면 그 시점에서 명령실행이 중단되였다가 후에 다시 복귀된다. 처리장치는 새치기가 발생하면 새치기발생명령보다 전에 있는 명령들이 이미 완료될수 있다는것을 복귀할 때 고려하여야 한다.

비순차실행, 비순차완료

순차실행을 리용하여 처리장치는 의존성 혹은 충돌이 일어 날 때까지 명령해신만을 할수 있다. 부가명령들은 충돌이 해제될 때까지 해신되지 않는다. 결과적으로 처리장치는 관흐름안에 이미 있는 명령과 독립이면서 관흐름에 들어 갈수 있는 다음명령과의 충돌점을 미리 예측할수 없다.

비순차실행을 실현하기 위하여서는 관흐름의 해신과 실행단계를 분리시켜야 한다. 이 분리는 **명령창문**이라고 하는 완충기로 실현된다. 이러한 구성에서는 처리장치가 명령에 대한 해신을 끝낸 다음 그것을 명령창문에 넣어 준다. 완충기가 완전히 차 있지 않는 한 처리장치는 새로운 명령에 대한 꺼내기와 해신을 계속할수 있다. 기능장치가 실행단에서 쓸수 있게 되였을 때 명령창문으로부터 나온 명령은 실행단에서 처리된다. 만일 그것이 리용할수 있는 부분기능장치를 요구하거나 충돌과 의존성이 이 명령을 차단하지 않으면 임의의 명령이 실행될수 있다. 이러한 구성체계를 가지므로 처리장치는 동시처리를 할수 있으며 실행단에 들어 올수 있는 독립적인 명령을 식별할수 있다. 명령들은 자기의 원천프로그람순서를 거의 무시하고 명령창문으로부터 출력된다. 이제 남은 문제는 프로그람

집행이 정확히 되는가 하는것이다.

그림 13-4 ㄷ는 이 규약을 보여 주고 있다. 매 주기마다 두개의 명령이 해신단에 입력된다. 매 주기는 완충기크기의 영향을 받게 되는데 두 명령은 해신단으로부터 명령창문에로 이동한다. 이 실례에서 15 에 앞서서 16 명령을 내보낼수 있다(15 는 14 에 의존하지만 16 은 의존하지 않는다는것을 다시 상기시킨다.). 따라서 명령과 재쓰기단에서 한주기가 절약되며 그림 13-4 ㄴ와 비교해 보면 끝에서 끝까지의 절약은 한주기이다.

그림 13-4 ㄷ에 명령창문의 역할을 보여 주었다. 그러나 이 창문은 관흐름단을 보충한것이 아니다. 창문안에 있는 명령은 처리장치가 명령을 관흐름출력할 때 요구되는 충분한 정보를 가지고 있다는것을 암시한다.

비순차관흐름출력, 비순차완료규약은 앞에서 서술한것과 같은 제약조건의 영향을 받는다. 명령은 의존성이나 충돌을 무시하면 관흐름출력를 할수 없다. 그 차이는 관흐름단들이 지연될수 있는 가능성을 줄임으로써 더 많은 명령을 출력할수 있게 한것이다. 그외에도 새로운 의존성이 제기되는데 그것이 앞에서 언급된 **반의존성**(혹은 **읽기—쓰기**의존성)이다. 앞에서 고찰한 코드부분이 이 의존성을 보여 주고 있다.

11 : R3 ← R3 op R5
12 : R4 ← R3 + 1
13 : R3 ← R5 + 1
14 : R7 ← R3 op R4

명령 13은 명령 12가 집행되기 시작하여 그의 연산수를 꺼내기전에는 실행될수 없다. 그 리유는 13 이 12 를 위한 원천연산수인 R3 을 갱신하기때문이다. 반의존성이라는 말이 쓰이게 된것은 이것이 실자료의존성과 류사하지만 실제는 반대이기때문이다. 즉 두번째 명령을 리용하기 위한 값을 만들어 내는 첫번째 명령대신에 두번째 명령이 첫번째 명령이 리용할 값을 파괴한다.

3. 등록기이름바꾸기

비순서명령관흐름출력이라든가 혹은 비순서명령완료가 허락되면 이것은 출력의존성과 반의존성이 일어 날수 있다. 이 의존성은 실자료의존성과 자원충돌과는 달리 프로그람을 통한 자료흐름과 실행순서에 영향을 준다. 다른 한편 출력의존성과 반의존성은 등록기안의 값들이 프로그람흐름에 의하여 받게 되는 값들의 순서에 더는 영향을 미칠수 없게 한다.

명령이 순서대로 관흐름출력되거나 순서대로 완료될 때 실행의 매점에서 매 등록기의 내용을 규정할수 있다. 비순차기술을 리용하면 프로그람에서 받게 되는 명령의 순차를 고려해야 하므로 제때에 매개 점에서의 등록기값을 충분히 알수 없다. 실제로 값들은 등록기리용으로 하여 충돌상태에 있게 되며 처리장치는 관흐름을 지연시켜 이 충돌을 풀어야 한다.

반의존성과 출력의존성은 모두 기억기충돌을 일으킨다. 다중명령들은 같은 등록기주소를 리용하기 위하여 경쟁하므로 성능제고를 방해하는 관흐름제약이 생긴다. 이 문제는 제 12 장에서 론의된것처럼 등록기최적화기술을 리용할 때 더 심각하다. 그것은 콤파일러기술이 등록기를 최대로 리용하려고 시도하며 이때 기억기충돌의 회수가 최대로 커지기때문이다.

이러한 류형의 기억기충돌을 극복하기 위한 한가지 방법은 전통적인 자원충돌해결법인 자원중첩법에 기초하고 있다. 여기에서는 이 기술을 **등록기이름바꾸기**라고 한다. 실제

로 등록기들은 처리장치하드웨어에 의하여 동적으로 배치되며 그것들은 임의의 시간에 여러점에서 명령에 요구되는 값들과 관련된다. 새로운 등록기값이 만들어 졌을 때(례하면 목적연산수로서 등록기를 가지는 명령을 실행할 때) 그 값을 새 등록기가 할당 받는다. 그 등록기안의 값을 원천연산수로서 호출하는 다음명령은 이름바꾸기처리를 하여야 하는데 이 명령들에서 등록기참조는 필요한 값이 들어 있는 등록기를 참조한것과 반대여야 한다. 따라서 여러개의 서로 다른 명령에서 같은 원천등록기참조는 서로 다른 값에 대하여 서로 다른 실지등록기를 참조하는것처럼 볼수 있다.

다음의 코드토막을 리용하여 등록기이름바꾸기가 어떻게 리용되는가를 보기로 하자.

```
11:   R3b ≧  R3a OP R5a
12:   R4b ≧  R3b + 1
13:   R3c ≧  R5a + 1
14:   R7b ≧  R3c op R4b
```

첨수가 없는 등록기참조는 명령안에서 찾게 되는 론리등록기참조를 의미한다. 첨수가 있는 등록기참조는 새로운 값을 취하기 위하여 할당된 장치등록기를 의미한다. 부분적인 론리등록기에 대하여 새롭게 할당되였을 때 그 론리등록기를 원천연산수로 하는 다음명령참조는 가장 최근(명령의 프로그람순차에 대한 견지에서 최근)에 할당된 하드웨어등록기를 참조하게 된다.

이 실례에서는 13 명령에서 등록기 R3c 를 만들어 냄으로써 두번째 명령에 대한 반의존성과 첫번째 명령에 대한 출력의존성을 피할수 있으며 이것은 14 에 의하여 호출되는 정확한 값과 충돌하지 않는다. 그 결과 13 은 즉시 실행될수 있다. 즉 이름바꾸기가 없으면 13 은 첫번째 명령이 완료되고 두번째 명령이 실행될 때까지 실행될수 없다.

4. 기계병렬화

앞에서 우리는 성능을 개선하기 위하여 슈퍼스칼라처리장치에서 리용될수 있는 3 가지 하드웨어기술인 자원의 중첩, 비순차실행, 이름바꾸기에 대하여 고찰하였다. 이 기술들사이의 관계를 보여 주는 한가지 연구내용이 참고문헌 [SHIT89]에 주었다. 이 연구는 모의를 리용하여 여러가지 슈퍼스칼라기능이 강화된 MIPS R2000 의 특성을 가진 기계를 모형화하였다. 여러개의 서로 다른 프로그람순서가 모의되였다.

그림 13-5 에 그 결과를 보여 주었다. 매개 그라프에서 수직축은 스칼라기계에 대한 슈퍼스칼라기계의 평균속도제고비에 대응한다. 수평축은 4 개의 서로 다른 처리장치의 구성에 대한 결과를 보여 주고 있다. 기준기계는 어떠한 기능장치도 모방하지 않았지만 명령을 비순차로 실행할수 있다. 두번째 점도표는 넣기/기억을 중첩하여 자료캐쉬를 호출한다. 세번째 도표는 산수-론리연산장치를 중첩하였으며 4 번째 도표는 넣기/기억산수-론리연산장치를 중첩하였다. 매개 그라프에서 결과는 8, 16, 32 개의 명령크기를 가진 명령창문에 대한것이며 처리장치가 할수 있는 예측량을 그림으로 보여 주었다.

두 그라프의 차이는 두번째 그라프에서 등록기의 이름바꾸기가 허락된것이다. 첫번째 그라프는 모든 의존성에 의하여 영향을 받는 기계를 반영한것이라고 말할수 있으며 두번째 그라프는 오직 실자료의존성에 의해서만 제한을 받는 기계에 대한것이다. 두 그라프를 통하여 몇가지 중요한 결과를 얻어 낼수 있다.

우선 등록기이름바꾸기가 없이 기능장치들을 보충하는것은 거의 의의가 없다. 그것은 성능개선이 거의 없고 하드웨어가 복잡해 짐에 따라 가격이 높아 지기때문이다. 이름바꾸기를 하여 반의존성과 출력의존성을 없애면 보다 더 많은 기능장치들을 보충하여 현

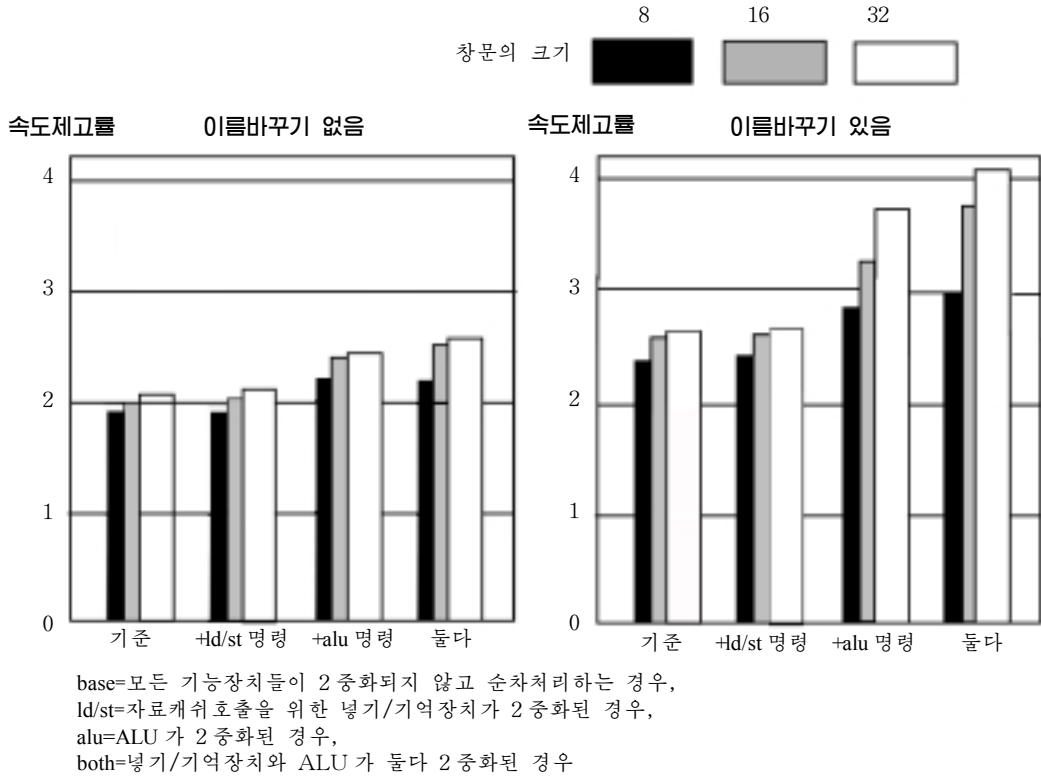



그림 13-5. 순차적인 의존관계가 없는 여러 처리장치구성들의 속도제고률

저한 리득을 얻을수 있다. 그러나 여기서 고려해야 할것은 8 개의 명령을 넣을수 있는 명령창문과 그보다 더 큰 명령창문을 리용하는것이 큰 차이가 있다는것이다. 그것은 명령창문이 너무 작으면 자료의존성으로 하여 많은 기능장치들을 효과적으로 리용할수 없기때문이다. 즉 하드웨어를 보다 더 충분히 리용하기 위하여 처리장치는 독립적인 명령을 찾기 위한 보다 앞선 탐색을 할수 있어야 한다.

5. 분기예측

고성능관흐름기계는 분기명령을 처리하기 위해 관흐름출력를 주소화하여야 한다. 례하면 Intel 80486 은 분기다음의 명령을 꺼내거나 분기하려는 명령을 꺼내는 방법으로 이 문제를 주소화하였다. 그러나 미리꺼내기와 실행사이에는 두개의 관흐름단이 있기때문에 이 방법은 분기가 생기면 두주기의 지연이라는 손해를 보게 된다.

RISC 기계의 출현과 더불어 지연분기방법이 개발되였다. 이 방법은 처리장치가 임의의 쓰지 않는 명령들을 미리꺼내기전에 조건분기명령의 결과를 타산해 두어야 한다. 이 방법을 리용함으로써 처리장치는 항상 분기다음에 오는 명령 하나를 실행한다. 이것은 처리장치가 새로운 명령흐름을 꺼내는동안 관흐름기능을 충분히 유지하게 한다.

슈퍼스칼라기계가 발전하면서 지연분기방법은 흥미가 적어 졌다. 그 리유는 다중명령을 명령의존성과 관계되는 일련의 문제를 발생시키는 지연슬로트안에서 실행하여야 하기때문이다. 그리하여 슈퍼스칼라기계는 선행 RISC 에서 쓰던 분기예측기술을 리용하였다.

PowerPC 601 과 같은 일부 기계들은 단순한 정적분기예측기술을 리용하고 있다. PowerPC 620 , Pentium II 와 같은 보다 개선된 처리장치들은 분기경력해석에 기초한 동적분기예측기술을 리용한다.

6. 슈퍼스칼라실행

여기서는 프로그람의 슈퍼스칼라실행에 대하여 개괄한다. 그림 13-6 에서 이것을 보여주었다. 실행되는 프로그람은 선형순차명령으로 되여 있다. 이것은 프로그람작성자에 의하여 작성되든가 콤파일러에 의하여 발생된 정적프로그람이다. 분기예측을 포함한 명령꺼내기처리는 동적명령흐름을 형성하는데 리용된다. 이 흐름은 의존성들을 검사하여 처리장치의 인공적인 의존성을 제거할수 있다. 그다음 처리장치는 명령들을 실행창문에 넣는다. 이 창문에서 명령들은 더는 순차성을 이루지 않지만 자기의 실자료의존성에 따라서 구조화된다. 처리장치는 실자료의존성과 하드웨어자원리용가능성에 의하여 결정된 순서에 따라 매개 명령의 실행단을 동작시킨다. 최종적으로 명령들은 순차적인 순서로 다시 놓이게 되며 그 결과들이 기록된다.

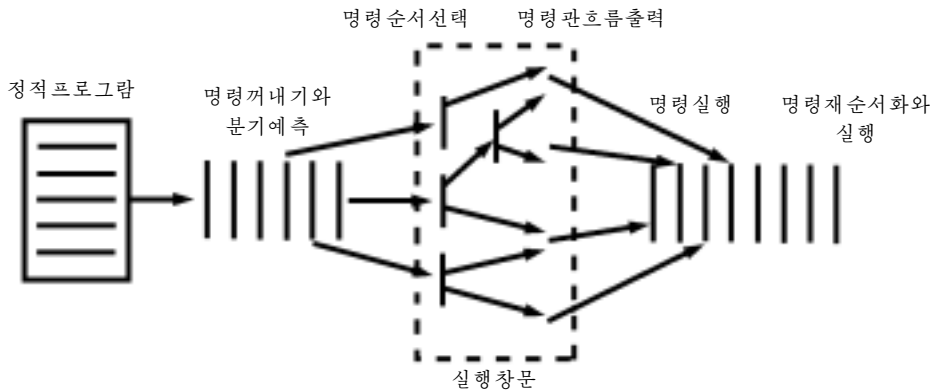



그림 13-6. 슈퍼스칼라처리에 대한 개념
(SMIT95)

이 마지막단계를 명령의 재구성이라고 한다. 이 단계는 다음과 같은 리유로 하여 필요하다. 병렬적이며 다중인 관흐름을 리용하기때문에 명령들은 정적프로그람에서 보여 준것과 다른 순서로 실행될수 있다. 더우기 분기예측과 추측실행이 리용됨으로써 일부 명령들은 실행을 완료할수 있으며 그때 그것들이 표현하는 분기가 취해 지지 않기때문에 금지되여야 한다. 그러므로 영구기억기와 프로그람작성자가 쓸수 있는 등록기들은 명령실행이 끝나는 즉시 갱신될수 없다. 결과들은 종속명령이 리용할수 있는 몇개종류의 림시기억기안에 넣어 지며 그다음 순차모형이 명령을 수행했다는것이 결정되였을 때에는 변하지 않는다.

7. 슈퍼스칼라실현

지금까지의 론의에 기초하면 슈퍼스칼라실현에 요구되는 처리장치하드웨어에 대한 몇가지 일반적인 해설을 할수 있다. [SMIT95]에서는 다음과 같은 주요요소를 제시하였다.

- 조건분기명령의 결과를 예측하거나 그것이 없이 꺼내기하는 방법으로 동시에 여러개의 명령을 꺼내는 명령꺼내기기술
  이 기능은 다중관흐름꺼내기 및 해신단들의 리용과 분기예측론리를 요구한다.
- 등록기값들을 포함한 실자료의존성을 결정하기 위한 론리와 실행하는데 필요한 위치에 이 값들을 전송하기 위한 기구
- 여러개의 명령을 병렬로 초기화하거나 관흐름처리하기 위한 기구
- 다중관흐름처리된 기능장치들과 다중기억참조를 동시에 봉사할수 있는 기억기의 계층구조를 비롯한 여러개의 명령을 병렬로 실행하기 위한 자원들
- 처리상태를 정확한 순서로 기억해 두기 위한 기구

제 3 절. Pentium Ⅱ

스칼라설계의 개념은 일반적으로 RISC 기본방식과 관련되지만 이와 같은 슈퍼스칼라원리를 CISC 기계에도 적용할수 있다. 이에 대한 가장 명백한 실례로는 Pentium Ⅱ를 들수 있다. Intel 계렬에서 슈퍼스칼라개념의 발전과정은 주목할만한 가치가 있다. 80486 은 슈퍼스칼라요소를 가지고 있지 않는 단순한 전통적인 CISC 기계이다. 초기에 나온 Pentium 은 두개의 개별적인 옹근수실행장치를 리용할수 있게 구성한 낮은 수준의 슈퍼스칼라요소로 되여 있었다. Pentium Pro 는 완전한 슈퍼스칼라설계를 도입하였다. Pentium Ⅱ는 MMX 실행장치가 보충된것외에 Pentium Pro 와 본질적으로 같은 슈퍼스칼라구성을 가지고 있다.

Pentium Ⅱ에 대한 일반적인 구성도를 그림 4-23 에 보여 주었다.

슈퍼스칼라구성에서 기본구성요소들은 명령꺼내기 및 해신장치, 명령선택 및 실행장치, 재구성장치이다. 이에 대하여 간단히 개괄한 다음 차례로 이 장치들에 대하여 보기로 한다.

IFU1	IFU2	IFU3	ID1	ID2	RAT	ROB	DIS	EX	RET1	RET2

IFU1 - 명령꺼내기장치　　ID - 명령해신
RAT - 등록기배치자　　ROB - 재순서완충기
DIS - 명령선택자　　EX - 실행단
RET - 명령재구성장치

그림 13-7. Pentium Ⅱ의 관흐름

Pentium Ⅱ의 조작과정을 종합하면 아래와 같다.

1. 처리장치는 정적프로그람의 순서에 따라 기억기로부터 명령을 꺼낸다.
2. 매개 명령은 하나 혹은 그이상의 고정길이 RISC 명령 즉 마이크로조작코드로 번역된다.
3. 처리장치는 마이크로연산이 비순차적으로 실행되도록 슈퍼스칼라관흐름조직에 따라 마이크로조작코드를 실행한다.
4. 처리장치는 매개 마이크로연산실행결과를 원천프로그람흐름의 순서대로 처리장치안의 등록기묶음에 기억한다.

결과적으로 Pentium II의 기본방식은 내부에 RISC 핵심을 가지고 외부에 CISC 쉘로 되여 있다. 내부의 RISC 마이크로조작코드는 최소한 11 단으로 된 관흐름을 통과한다(그림 13-7): 일부 경우에 마이크로연산은 다중실행단을 요구한다. 이것은 Intel x86 처리장치들과 Pentium에서 리용되는 5단(그림 11-18)으로 된 관흐름과 대조를 이룬다.

1. 명령꺼내기와 해신장치

그림 13-8은 Pentium II의 명령꺼내기와 해신장치를 간단히 보여 준것이다. 꺼내기조작은 3 개의 관흐름단으로 구성되여 있다. 우선 IFU1 단은 명령캐쉬로부터 한번에 한행(32byte)씩 명령을 꺼낸다. NEXT-IP 장치는 꺼내려는 다음명령의 주소를 준비한다. 캐쉬행에는 IFU1 완충기에 넣어 야 할 명령이 들어 간다. 이것을 결정하는것은 지시기를 증가하는 식으로 쉽게 계산되지 않는다. 그것은 지시기를 다른 주소로 변화시키는 분기조작이나 새치기가 있을수 있기때문이다.

다음으로 IFU1 완충기의 내용은 한번에 16byte씩 IFU2에 넘겨 진다. 이 장치는 두개조작을 병렬로 수행한다. IFU2 는 명령의 경계를 결정하기 위하여 그 바이트들을 주사한다. 즉 이것은 Pentium의 명령이 가변길이기때문에 반드시 필요한 조작이다. 이 명령들가운데

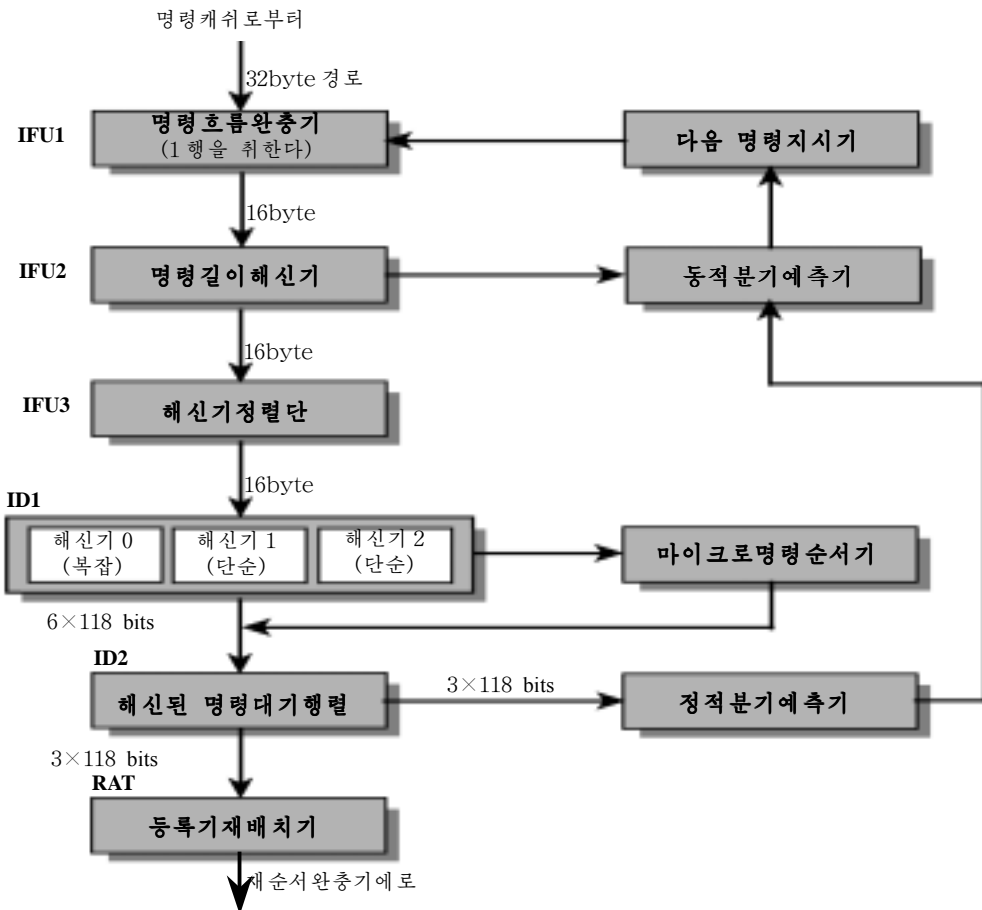



그림 13-8. Pentium II의 꺼내기/해신장치

서 일부가 분기명령이라면 그 장치는 동적분기예측기에 대응하는 기억주소를 넘긴다. 그 다음 IFU2는 IFU3에 16byte로 된 블로크를 넘긴다. 여기서 IFU3은 해당한 해신기에서 리용할 명령을 배렬하는 기능을 수행한다.

IFU3의 조작을 리해하자면 명령해신기의 첫단인 ID1을 알아야 한다. 이 단은 3개의 Pentium II 기계명령을 병렬로 조종할수 있다. ID1은 모든 기계어명령을 각각 118bit RISC 명령으로 된 1～4개의 마이크로조작코드로 변환한다. 비교할 때 주의하여야 할것은 가장 단순한 RISC 기계는 명령길이가 32bit로 되여 있다는것이다. 보다 복잡한 Pentium 조작을 하기 위하여서는 보다 긴 마이크로조작코드길이가 요구된다. 그럼에도 불구하고 마이크로연산은 그것을 발생시키는 원천명령보다 관리하기가 쉽다. ID1은 3개의 해신기로 되여 있다. 이들중에서 첫번째것은 Pentium명령을 조종하여 4개까지의 마이크로연산코드로 변환한다. 두번째와 세번째해신기들은 하나의 마이크로조작코드로 변환되는 단순한 Pentium 명령을 조종하는데 이 명령들은 등록기—등록기명령들로 되여 있다. ID 1 의 구조에 적응시키기 위하여 IFU3은 필요하다면 16byte로 된 완충기안의 첫번째 명령이 복합명령이고 그다음의 두 명령이 단순명령이 되도록 그 완충기의 내용을 회전시킨다. 만일 세 명령이 다 단순명령이면 회전조작이 필요 없다. 여러개의 명령이 복합명령이면 그때 그 명령들은 두번째와 세번째해신기들이 복합명령으로 표현되지 않도록 단안에 ID1로 있어야 한다.

극히 일부 명령들이 4 개이상의 마이크로연산을 요구하는 경우가 있다. 이 명령들은 마이크로명령순서기(MIS)에로 전송되는데 이 순서기는 복잡한 기계어명령과 관련된 마이크로조작코드의 렬(5 혹은 그이상)이 들어 가는 마이크로코드 ROM 을 말한다. 례하면 문자렬명령은 대단히 많고(심지어 수백개) 반복적인 마이크로코드순서로 변환된다. 따라서 마이크로명령순서기(MIS)는 제 4편에서 론의된 마이크로프로그람화된 장치이다.

ID1 혹은 MIS의 출력은 한번에 6개의 마이크로조작코드까지 블로크화할수 있는 해신장치 ID2의 두번째 단에 입력된다. ID2는 원천프로그람의 순서로 마이크로조작코드의 행렬을 짓는다. 여기에서 두번째 분기예측을 하게 된다. 만일 조작코드가운데서 분기가 있으면 이것들을 정적분기예측장치에 넘겨 주며 그다음 동적분기예측장치로 간다. 분기예측에 대하여서는 이 절의 마지막에 서술하였다.

ID2 안에 순차배렬된 마이크로조작코드는 등록기할당기(RAT)로 알려 진 등록기과정을 통과한다. RAT 는 16 개의 방식등록기(8 개의 류점수등록기들과 EAX, EBX, ECX, EDX, ESI, EDC, EBP, ESP)에 대한 참조를 40 개의 물리등록기참조로 재배치한다. 이 단은 실자료의존성(쓰기후 읽기)을 유지하는 동안 방식등록기의 수가 제한된것으로 하여 생기는 거짓의존성을 제거한다. 그다음 RAT는 수정된 마이크로조작코드를 재순서완충기(ROB)에 넣는다.

2. 재순서완충기

재순서완충기(ROB)는 40개까지의 마이크로조작코드를 취할수 있으며 또한 40개의 하드웨어등록기로 된 순환완충기이다. 매개 완충기의 머리부는 다음과 같은 마당으로 구성되여 있다.

- **상태마당**: 이 마당은 마이크로 조작코드가 실행순서로 되여 있는가, 실행을 위하여 선택되였는가, 실행이 완성되였는가, 재구성할 준비가 되여 있는가를 지적한다.
- **기억주소**: 마이크로조작코드에 의하여 발생된 Pentium 명령의 주소, 실행을 위하여 선택되였는가, 실행이 완성되였는가, 재구성할 준비가 되여 있는가를 지적한다.
- **마이크로조작코드**: 실제의 조작

- **별명등록기**: 만일 마이크로조작이 16개의 방식등록기들가운데서 어느 하나를 참조한다면 이 머리부는 40개의 하드웨어등록기가운데서 하나를 참조하게 재조종된다.

마이크로조작코드는 차례로 ROB에 들어 간다. 그다음 마이크로조작코드는 ROB부터 선택되여 선택 및 해신장치에 넘어 간다. 선택을 위한 기준은 이 마이크로조작을 위해 요구되는 적합한 실행장치와 필요한 자료항목이 사용될수 있는 경우이다. 마지막으로 마이크로조작코드는 차례로 ROB로부터 출력된다. 순차적인 재구성을 실현하기 위하여 마이크로조작코드는 매개 마이크로조작코드가 출력을 목적으로 지적된 다음 제일 오랜것을 먼저 내보낸다.

3. 선택 및 실행장치

그림 13-9에서 Pentium의 선택 및 실행장치를 간단히 보여 주었다. 예약부는 ROB로부터 마이크로코드를 받으며 실행을 위하여 이것을 선택하여 결과를 다시 ROB에 기억시

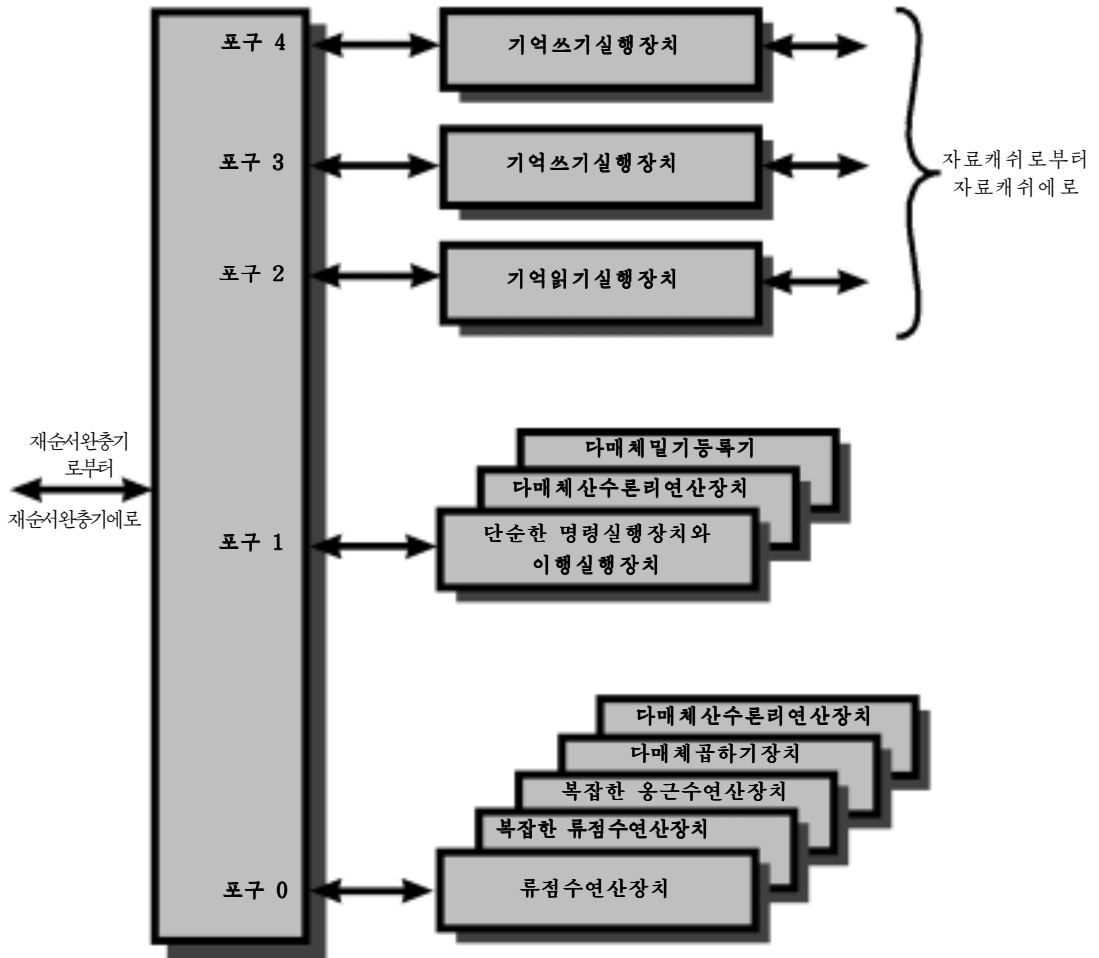



그림 13-9. Pentium II의 명령순서선택 및 실행장치

키는 기능을 수행한다. RS는 ROB안에서 마이크로조작코드를 찾으며 여기서 상태는 마이크로조작코드가 자기의 연산수를 모두 가지고 있다는것을 나타낸다. 만일 그 마이크로조작코드에 의하여 필요되는 실행장치가 연산가능하다면 그때 RS는 마이크로조작코드를 꺼내기 위하여 그것을 적당한 실행장치에로 즉시에 보낸다. 한주기동안에 5개의 마이크로조작코드까지 전송할수 있다. 여러개의 마이크로조작코드가 주어 진 실행장치를 리용할수 있다면 그때 RS는 그것들을 ROB로부터 차례로 내보낸다. 이것은 순차로 명령을 집행하는 일종의 FIFO(선입선출관흐름)이지만 이때에 명령흐름은 실질적으로 비순차적인 의존성과 분기에 의하여 재배렬된다.

다섯개의 포구는 RS를 다섯개의 실행장치와 련결시킨다. 포구 0은 포구 1에 할당되여 있는 단순옹근수연산외에 잘못된 분기예측의 조종기능을 가진 옹근수 및 류점수연산에 리용된다. 더우기 MMX실행장치는 이 두 포구사이에 놓인다. 나머지포구들은 기억기읽기쓰기용이다.

일단 실행이 끝나면 ROB 안에 있는 해당한 머리부는 갱신되므로 실행장치는 다른 마이크로조작을 할수 있다.

선택 및 실행장치를 통한 마이크로조작의 정확한 방향은 분기예측을 잘못하면 파괴될수 있다. 만일 분기예측이 오유로 하여 달라 지면 관흐름으로부터 없애 버려야 할 마이크로조작이 여러개의 처리단안에 있게 된다. 이것은 이행실행장치가 담당한다. 분기가 실행될 때 분기결과는 예측하드웨어가 예측하는것과 다르게 비교된다. 만일 비교가 일치하지 않으면 그때 JEU 장치는 그것들을 명령공유기억기로부터 제거하기 위하여 분기뒤에 놓여 있는 모든 마이크로조작코드의 상태를 바꾸어 놓는다. 그다음 분기예측을 위해 적당한 분기목적지(분기하여 가려는 곳)를 제공하며 새로운 목적주소로부터 전체 관흐름을 다시 시작한다.

4. 재구성장치(RU)

이 장치는 명령실행의 결과를 보관하는 재순서완충기를 관리한다. 먼저 RU 는 잘못된 분리예측과 집행되였지만 선행분기처리가 아직 유효하지 않은 마이크로조작을 고려하여야 한다. 일단 마이크로조작코드가 집행되고 잘못된 분기예측에 의해 동작에 영향이 없다면 그것은 재구성을 위한 준비로 된다. 선행한 Pentium명령이 재구성되고 그다음명령의 연산조작코드전체가 동작완료를 위한 준비로서 표식되였다면 RU는 이 명령에 의하여 영향을 받은 방식등록기들을 모두 갱신하여 ROB로부터 마이크로조작코드를 삭제한다.

5. 분기예측

Pentium Ⅱ는 분기명령의 실행에 기초한 동적분기예측방식을 리용하고 있다. 분기목적완충기(BTB)는 가장 최근에 만난 분기명령에 대한 정보를 기억하여 둔다. 분기명령이 명령흐름속에 나타났을 때 BTB 를 검사한다. 만일 머리부가 BTB 안에 이미 존재하면 명령장치는 분기를 예측하겠는가 안하겠는가를 결정하는 그 머리부에 대한 경력정보에 따라 동작하게 된다. 만일 분기가 예측되면 이 머리부와 관련되여 있는 분기목적주소는 분기목적명령을 미리 꺼내는데 리용된다.

일단 명령이 실행되면 해당한 머리부의 경력부분은 갱신되여 분기명령실행결과에 영향을 준다. 만일 이 명령이 BTB안에 없으면 이 명령의 주소는 BTB안의 머리부에 넣어진다. 필요하다면 오랜 머리부는 삭제된다.

앞의 두절은 Pentium Ⅱ와 Pentium Pro 와 같은 Pentium 이 리용되는 분기예측방식을 설명하였다. 그러나 Pentium 의 경우에 상대적으로 단순한 2bit 경력방법이 리용된다. Pentium Pro 와 Pentium Ⅱ는 훨씬 더 긴 관흐름길이(Pentium 이 5 단위인데 비하여 11 단이상의 단)를 가지며 그러므로 잘못된 예측에 대한 성능은 더 떨어 진다. 따라서 Pentium Pro 와 Pentium Ⅱ는 잘못된 예측률을 줄이기 위하여 보다 더 긴 경력비트를 가진 보다 정교한 분기예측방법을 리용한다.

Pentium Ⅱ의 BTB 는 512 행을 가진 4 통로묶음련상캐쉬로 되여 있다. 매 머리부는 분기주소를 꼬리표로 리용한다. 머리부는 또한 이 분기가 취해 진 마지막시간에 따르는 분기목적주소들과 4bit 의 경력마당으로 되여 있다. 따라서 4 개의 경력비트를 리용하는것은 처음에 나온 Pentium 과 슈퍼스칼라처리장치에서 리용된 2bit 와 대조를 이룬다. 4bit 를 리용함으로써 Pentium Ⅱ기구는 분기예측에 보다 더 오랜 경력을 고려해 둘수 있다. 흔히 리용되는 알고리듬으로서 예흐알고리듬이라고 부르는것이 있다[YEH91]. 이 알고리듬의 개발자들은 오직 2bit 로 된 경력을 리용하는 알고리듬과 비교하면 잘못된 예측이 크게 작아진다는것을 증명하였다[EVER91].

BTB 안의 경력이 없는 조건분기명령은 다음과 같은 규칙에 따르는 정적예측알고리듬을 리용하여 예측한다.

- 상대 IP 가 없는 분기주소인 경우 분기가 복귀하면 예측이 취해 지며 그 반대인 경우는 예측이 취해 지지 않는다.
- 상대 IP 역방향조건분기인 경우 예측이 취해 진다. 이 규칙은 전형적인 순환동작에 영향을 미친다.
- 상대 IP 정방향조건분기인 경우 예측이 취해 지지 않는다.

제 4 절. PowerPC

PowerPC 기본방식은 IBM 801 , RT PC 그리고 RS/6000 과 같은 계렬인데 POWER 기본방식이 실현된것이라고 말할수 있다. 이것들은 RISC 기계이지만 이 계렬안에서 첫번째것은 슈퍼스칼라특징을 가지고 있으며 그것이 RS/6000 이다. PowerPC 기본방식의 첫 실현인 601 은 RS/6000 과 매우 류사한 슈퍼스칼라실례이다. 그다음에 나온 PowerPC 모형은 보다 더 슈퍼스칼라개념에 접근하고 있다. 이 절에서는 RISC 에 기초한 슈퍼스칼라설계에 아주 적합한 PowerPC 601 에 대하여 보기로 한다.

1. PowerPC 601

그림 13-10 에 601 구성에 대한 일반적고찰을 주었다. 다른 슈퍼스칼라기계에 비하면 601 은 실행을 더 많이 중첩시키기 위하여 기능장치들을 독립적으로 구성하였다. 특히 601 의 핵심부는 3 개의 독립적인 관흐름실행장치들 즉 옹근수장치, 류점수장치, 분기처리장치들로 구성되여 있다. 이 장치들은 공동으로 한번에 3 개의 명령을 실행할수 있다.

그림 13-11 에 기능장치들사이의 명령흐름을 강조하기 위하여 601 의 기본방식의 론리적고찰을 보여 주었다. 꺼내기장치는 캐쉬로부터 한번에 8 개의 명령까지 미리 꺼낼수 있다. 캐쉬는 결합된 명령 및 자료캐쉬를 지원하며 명령을 다른 장치에 넣어 주거나 자료를 등록기에 넣어 주는 기능을 수행한다. 캐쉬중재론리는 제일 높은 호출우선권을 가진 주소를 캐쉬에 보낸다.

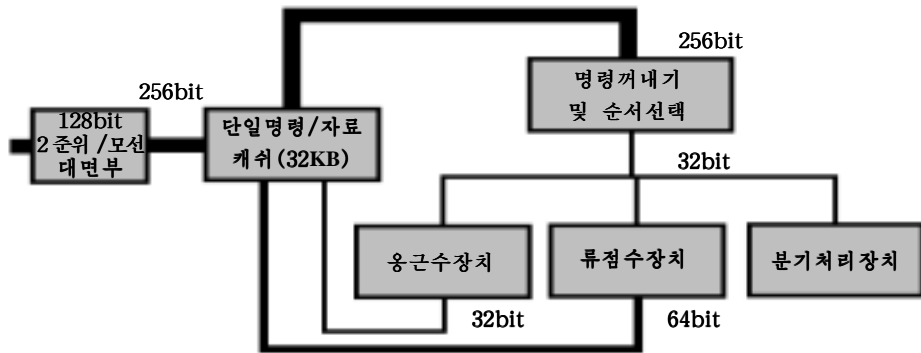



그림 13-10. PowerPC 601의 블로크도

선택장치

선택장치는 캐쉬로부터 명령을 꺼내여 선택대기렬에 넣어 주는데 한번에 8개의 명령을 취할수 있다. 명령의 안전한 흐름을 분기처리장치, 옹근수장치, 류점수장치에 주기 위하여 이 명령의 흐름을 처리한다. 대기렬의 웃절반은 명령이 아래절반에로 옮겨 갈 때까지 유지하는 완충기기능을 수행한다. 그 목적은 선택장치가 캐쉬로부터 명령이 나오는 시간까지 기다리면서 지연되지 않는가를 확인하자는데 있다. 아래절반에서 명령들은 다음과 같은 방식에 따라 선택된다.

- **분기처리장치:** 모든 분기명령을 조종한다. 명령선택대기렬의 아래절반의 제일 아래 명령은 이 장치가 그것을 접수할수 있다면 분기처리장치에로 넘어 간다.
- **류점수연산장치:** 모든 류점수명령을 조종한다. 명령선택렬의 아래절반의 제일아래에 있는 명령은 그 장치안에 명령관흐름이 차 있지 않으면 류점수장치에로 실행된다.
- **옹근수연산장치:** 옹근수명령들은 옹근수명령, 등록기파일과 캐쉬사이의 읽기, 쓰기, 옹근수비교명령을 조종한다. 옹근수명령은 그것이 선택렬의 아래에 넣어 진(채워진)후에야 관흐름장치로부터 출력된다.

분기와 류점수명령이 선택렬로부터 순서가 다르게 출력되기때문에 분기처리장치와 류점수장치안의 명령관흐름들이 계속 충만되게 하며 그것은 될수록 빨리 명령을 선택대기렬을 통하여 이동하게 할수 있다.

선택장치는 또한 미리 꺼내야 할 주소를 계산할수 있는 론리장치를 가지고 있다. 이것이 있어 분기명령이 선택대기렬의 아래절반으로 들어 갈 때까지 명령꺼내기를 차례로 계속할수 있다. 분기처리장치가 한 명령을 처리할 때 뒤따르는 명령들을 새로운 주소로부터 꺼내여 선택대기렬에 넣어 지도록 다음꺼내기주소를 갱신할수 없다.

명령관흐름

그림 13-12는 여러 장치에 따르는 명령관흐름을 보여 준다. 모든 명령에 대하여 공통인 꺼내기주기가 있는데 이것은 명령이 특별한 장치에서 선택되기전에 일어 난다. 두번째 주기는 특별한 장치에 대한 명령의 선택을 진행한다. 이것은 그 장치안의 다른 동작과 중첩된다. 매개 박자주기동안 명령선택장치는 명령대기렬의 아래의 4개 머리부를 고려하여 그가운데서 3개 명령까지 선택한다.

분기명령인 경우 두번째 주기기간 분기예측과과 같이 명령의 해신과 실행을 한다. 마

지막동작은 다음절에서 론의하게 된다.

옹근수장치는 류점수읽기쓰기를 비롯하여 기억기의 읽기/쓰기조작, 등록기－등록기옮김 혹은 산수-론리연산조작을 일으키는 명령을 취급한다. 읽기/쓰기인 경우에는 결과주소를 캐쉬에 보내는 주소발생주기, 되쓰기주기가 있다.

류점수명령은 류사한 관흐름에 의하여 처리되지만 실행주기가 두개이며 류점수연산의 복잡성을 반영하고 있다.

몇가지 보충적인 문제들에 주의를 돌려야 한다. 조건등록기는 8 개의 독립적인 4bit 조건코드마당을 가지고 있다. 조건등록기는 여러개의 조건코드들을 등록하고 있는데 이로

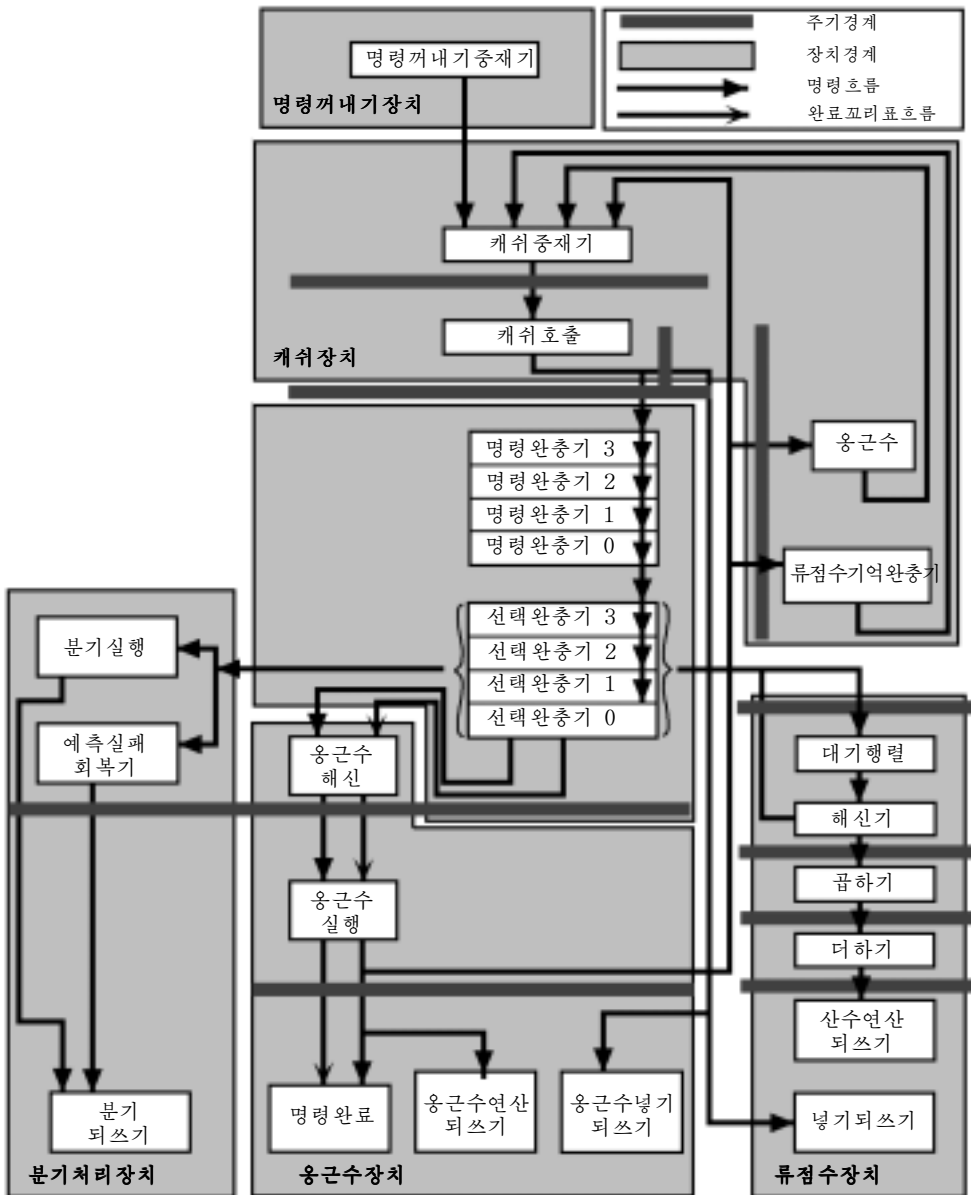



그림 13-11. PowerPC 601 의 관흐름구조[POTT 94]

하여 호상관건이나 명령들사이의 의존성을 줄일수 있다. 례하면 콤파일러는 다음의 과정으로 변환할수 있다

```
compare
branch
compare
branch
…

compare
compare
…
branch
branch
…
```

왜냐하면 매개 기능장치는 그 조건코드를 조건등록기안의 서로 다른 마당에 보낼수 있기때문에 조건코드의 공유에 의하여 생기는 명령들사이의 호상관건을 피할수 있다.

분기처리장치안에 보관 및 기억등록기(SRRs)가 있으므로 다른 기능장치들에 론리기능을 더 보충하지 않고도 단순한 새치기들과 쏘프트웨어새치기들을 조종할수 있다. 따라서 단순한 조작체계봉사는 기능장치들사이의 복잡한 상태조종이나 동기화가 없이 빠르게 실현될수 있다.

601 은 비순차적으로 분기와 류점수명령을 실행할수 있기때문에 적합한 실행을 담보하기 위한 조종이 필요하다. 의존성이 존재할 때(례하면 명령이 선행명령에 의하여 아직 처리되여야 할 연산수를 가지고 있을 때) 대응하는 장치의 관흐름은 지연된다.

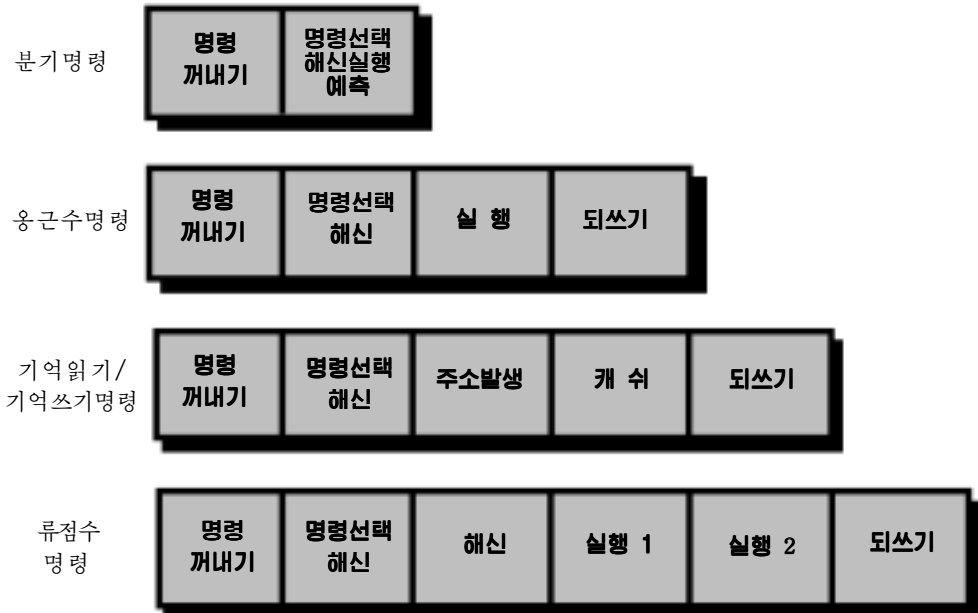



그림 13-12. PowerPC 601 관흐름명령

2. 분기처리

RISC 나 슈퍼스칼라기계의 성능을 높이기 위한 기본수법은 관흐름의 리용을 최적화하는것이다. 이 설계에서 제일 문제로 되는 요소는 분기조종을 어떻게 하는가 하는것이다. PowerPC 에서는 분기처리를 분기장치가 담당한다. 이 장치는 많은 경우에 분기가 다른 장치들의 실행걸음에 영향을 미치지 않도록 설계되였으며 이러한 류형의 분기를 령주기분기라고 한다. 령주기분기를 실현하기 위하여 다음과 같은 방안을 리용하고 있다.

1. 분기를 위하여 선택완충기를 주사하기 위한 론리회로가 제공된다. 분기가 대기렬의 아래절반에 처음 나타나고 선행분기가 없어서 실행이 미정으로 있을 때 분기목적주소가 생긴다.
2. 조건분기의 결과를 결정한다. 조건코드가 충분히 설정되였다면 분기를 결정할수 있다. 어떤 경우에는 분기명령이 나타나자마자 론리회로가 분기를 결정한다.

   ㄱ) 분기를 취할수 있는 경우; 이것은 조건코드가 주어 지고 분기를 지적하는 조건분기나 무조건분기인 경우이다.

   ㄴ) 분기가 취해 질수 없는 경우; 이것은 조건코드가 주어 지고 분기가 없다는것을 지적하는 조건분기인 경우에 해당된다.

   ㄷ) 출력을 아직 결정할수 없는 경우는 분기가 역방향분기(표준적으로 순환)를 취할수 있고 정방향을 취할수 없는것으로 추측될 때이다. 분기명령다음의 명령들은 조건방식에 따라 실행장치에 들어 간다. 일단 조건코드값이 실행장치안에서 만들어 지면 분기장치는 관흐름안의 명령을 삭제하고 꺼낸 목적주소에 따라 처리를 진행하든가 조건명령에 대한 신호들을 지운다. 콤파일러는 이 표준기동을 예약하는데 명령코드로 한비트를 리용할수 있다.

분기경력에 기초한 분기예측기술은 그리 좋은 방안이 못되므로 받아 들이지 않고 있다.

분기예측의 효과에 대한 실례로서 그림 13-13 에 제시된 프로그람을 고찰해 보면 분기처리장치가 조건분기명령이 취해 지지 않는다는것을 예측할수 있다(정방향분기의 표준실례). 그림 13-14 ㄱ의 실례는 분기가 취해 지지 않는 경우의 관흐름효과에 대하여 보여 주었다. 첫번째 주기에서 선택대기렬에는 8 개의 명령이 넣어 진다. 첫 6 개의 명령은 옹근수명령이며 한주기당 1 개의 명령이 옹근수장치에 들어 가게 된다. 조건분기명령은 그것이 선택대기렬의 아래절반에 갈 때까지 선택될수 없으며 이것은 5 번째 주기에서 일어 난다. 분기장치는 이 분기가 취해 지지 않을것이라는것과 다음명령이 조건적으로 선택된다는것을 예측한다(D'로 표시된다). 분기는 비교명령이 8 번째 주기에서 실행될 때까지 변하지 않는다. 그때에 분기처리장치는 그 예측이 정확했고 실행이 계속되고 있다는것을 확인한다. 여기에서는 지연이 없으며 관흐름은 한 상태로 있다.

주목할것은 주기 4 로부터 주기 8 까지의 기간 명령꺼내기를 할수 없다는것이다. 그 리유는 캐쉬넣기가 5 개의 넣기명령이 들어 가는 캐쉬호출단으로 하여 2 주기동안 동작상태에 있기때문이다. 또한 명령흐름은 지연되지 않으며 그로 하여 선택대기렬은 8 개의 명령을 취할수 있다.

그림 13-14 은 예측이 부정확할 때 분기가 취해 지는 경우의 결과를 보여 준다. 이 경우에 IF 에서 시작하는 3 개의 명령을 상기(다시읽기)하며 ELSE 에서 시작하는 명령에서 꺼내기를 다시 시작한다. 결과적으로 옹근수관흐름의 실행단은 주기 9 와 주기 10 기간 무효이며(아무 조작도 하지 않는다.) 따라서 부정확한 예측으로 하여 두주기는

손실되게 된다.

3. PowerPC 620

620 은 PowerPC 기본방식에서 처음으로 64bit 를 실현하였다.

새로운 G3 과 같은 이 처리장치의 일반골격도를 그림 4-25 에 주었다. 이 실현에서 주목할 특징은 6 개의 독립적인 실행장치를 가지고 있는것이다.

- 명령장치
- 3 개의 옹근수장치
- 읽기/쓰기장치
- 류점수장치

이 기본방식은 처리장치가 세개의 옹근수장치들과 하나의 류점수장치에 동시에 4 개의 명령을 전송할수 있는것이다.

```
if  ( a > 0 )
        a = a + b + c + d + e ;
else
        a = a - b - c - d - e ;
```

ㄱ)

```
                                        # r1  points  to  a ,
                                        # r1 + 4  points  to  b ,
                                        # r1 + 8  points  to  c ,
                                        # r1 +12  points  to  d ,
                                        # r1 +16  points  to  e ,
        lwz     r8 = a ( r1)            # load  a
        lwz     r12 = b ( r1, 4 )       # load  b
        lwz     r9 = c ( r1, 8 )        # load  c
        lwz     r10 = d ( r1, 12 )      # load  d
        lwz     r11 = e ( r1, 16 )      # load  e
        cmpi    cr0 = r8, 0             # compare  immediate
        bc      ELSE, cr0 / gt = false  # branch  if  bit  false

IF:     add     r12 = r8, r12           # add
        add     r12 = r12, r9           # add
        add     r12 = r12, r10          # add
        add     r4 = r12, r11           # add
        stw     a ( r1 ) = r4           # store
        b       OUT                     # unconditional  branch

ELSE:   subf    r12 = r12, r8           # subtract
        subf    r12 = r9, r12           # subtract
        subf    r12 = r10, r12          # subtract
        subf    r4 = r12, r11           # subtract
        stw     a ( r1 ) = r4           # store

OUT:
```

ㄴ)

그림 13-13. 조건분기를 가지는 프로그람실례
ㄱ-C 언어, ㄴ-기계어

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	lwz	r8 = a (r1)	F	D	E	C	W											
	lwz	r12 = b (r1, 4)	F	·	D	E	C	W										
	lwz	r9 = c (r1, 8)	F	·	·	D	E	C	W									
	lwz	r10 = d (r1, 12)	F	·	·	·	D	E	C	W								
	lwz	r11 = e (r1, 16)	F	·	·	·	·	D	E	C	W							
	cmpi	cr0 = r8, 0	F	·	·	·	·	·	D	E								
	bc	ELSE, cr0 / gt = false	F	·	·	·	S											
IF:	add	r12 = r8, r12	F	·	·	·	·	·	·	D	E	W						
	add	r12 = r12, r9			F	·	·	·	·	·	D	E	W					
	add	r12 = r12, r10			F	·	·	·	·	·	·	D	E	W				
	add	r4 = r12, r11									F	·	D	E	W			
	add	a (r1) = r4									F	·	·	D	E	C		
	b	OUT																
ELSE:	subf	r12 = r12, r8																
	subf	r12 = r12, r9																
	subf	r12 = r12, r10																
	subf	r4 = r12, r11																
	stw	a (r1) = r4																
OUT:																		

ㄱ)

			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	lwz	r8 = a (r1)	F	D	E	C	W											
	lwz	r12 = b (r1, 4)	F	·	D	E	C	W										
	lwz	r9 = c (r1, 8)	F	·	·	D	E	C	W									
	lwz	r10 = d (r1, 12)	F	·	·	·	D	E	C	W								
	lwz	r11 = e (r1, 16)	F	·	·	·	·	D	E	C	W							
	cmpi	cr0 = r8, 0	F	·	·	·	·	·	D	E								
	bc	ELSE, cr0 / gt = false	F	·	·	·	S											
IF:	add	r12 = r8, r12	F	·	·	·	·	·	·	D								
	add	r12 = r12, r9			F	·	·	·	·	·								
	add	r12 = r12, r10			F	·	·	·	·	·								
	add	r4 = r12, r11																
	add	a (r1) = r4																
	b	OUT																
ELSE:	subf	r12 = r12, r8									F	D	E	W				
	subf	r12 = r12, r9									F		D	E	W			
	subf	r12 = r12, r10									F			D	E	W		
	subf	r4 = r12, r11									F				D	E	W	
	stw	a (r1) = r4									F					D	E	C
OUT:																		

ㄴ)

F - 꺼내기　　C -캐취호출　　D - 선택/해신
W - 되쓰기　　E - 실행/주소　　S - 선택

그림 13-14. 분기예측:분기예측기능이 없는 경우[WEIS94]
ㄱ-정확한 예측: 분기되지 않는 경우, ㄴ-정확하지 않은 예측: 분기되는 경우

620 은 고성능분기예측기술을 리용하고 있는데 그것은 실행장치안에 예측론의, 등록기이름고치기완충기, 예약부를 가지고 있기때문이다. 명령이 꺼내질 때 등록기가 기억하는것처럼 명령결과를 림시로 보관하기 위하여 개명완충기를 할당한다. 개명완충기를 리용하기때문에 처리장치는 분기예측에 기초한 명령을 투기적으로 실행할수 있다. 만일 예측이 부정확한것이 판명되면 그때 투기명령의 결과는 등록기파일을 파괴함이 없이 상기될수 있다. 일단 분기의 결과가 일치하면 림시결과는 영구적으로 써넣을수 있다.

매개 장치는 둘이상의 예약부를 가지고 있는데 이것은 선택된 명령을 기억하며 다른 명령들의 결과를 유지한다. 이런 특징은 명령장치에서 이 명령들을 명백히 하며 이 장치들이 다른 실행장치에 명령을 계속 보낼수 있게 한다.

620은 4개의 분해되지 않은 분기명령까지 투기적으로 집행할수 없다. 분기예측은 2048개의 머리부로 된 분기경력표의 리용에 기초한다. PowerPC 설계자들에 의해 진행된 모의실행결과는 분기예측성공률이 90%라는것을 보여 주고 있다[THOM94].

제 5 절. MIPS R10000

MIPS R4000 에서 기원하여 명령모임이 같은 MIPS R10000 은 슈퍼스칼라설계원리를 더 명백하고 간단하게 실현하였다. 그림 13-15 에 그의 총적구조를 보여 주었다.

R10000 명령처리의 첫단은 명령미리해신단이다. 미리해신은 PowerPC 620 과 Ultra Sparc 를 비롯하여 일련의 슈퍼스칼라기계들에서 찾아 볼수 있는 기능이다. 미리해신의 목적은 병렬실행판흐름의 요구를 만족시키는데 있다. 일부 슈퍼스칼라기계에서 의존성과 분기시험을 포함한 해신-실행통로는 높은 명령처리능력실현에서 중요하다. 그것은 이 통로를 통하여 명령들이 전달되여 다중실행장치들에로 분리되기때문이다. 따라서 이 통로를 잠재적인 문제점으로 보고 있다. 해신-꺼내기통로의 속도를 높이기 위하여 일부 슈퍼스칼라처리장치들은 그것들의 외부에 있는 2 차 캐쉬로부터 소자내부의 명령캐쉬에로 읽기하는것처럼 명령을 부분적으로 해신한다. 미리해신은 들어 오는 명령을 분류하며 뒤의 처리를 간단하게 하기 위하여 매개 명령에 몇개의 해신비트들을 추가한다. R1000 의 경우에 매개 명령에 4 개의 해신비트가 추가되여 이 명령을 선택할 실행장치를 지적한다.

R10000 은 2 차캐쉬로부터 한번에 4개의 명령을 꺼내며 그것들은 미리 해신한 다음 명령캐쉬에 넣어 진다. 다른 슈퍼스칼라기계들과 마찬가지로 꺼내려는 명령의 수는 최소한 해신 및 실행속도의 첨수값에 맞추도록 설계된다. 그러므로 R10000 은 병렬로 명령을 4 개까지 해신한다. 무조건분기가 나타나면 그때 늘 다음꺼내기를 위한 지적자를 조종하기 위하여 명령캐쉬에 분기목적지를 되돌려 준다. 조건분기가 나타나면 해신단은 2bit 의 경력방법을 리용하여 분기예측을 수행하는데 이때 명령들은 예측통로를 통하여 투기적으로 꺼내게 된다.

그다음 단은 등록기이름을 바꾸는 단으로서 5bit 의 론리주소번호(그림 12-8)가 배치되여 있다. 해신단은 론리주소를 물리주소에로의 배치와 리용중에 있거나 리용할수 있는 등록기목록을 유지한다.

해신한 다음 R10000 은 매개명령을 3개의 명령페지렬가운데서 어느 하나에 넣는다. 매개 대기렬은 16 개 명령까지 취할수 있다. 명령이 선택되여 예약비트는 명령안에 이름이 규정된 물리적결과등록기를 동작상태로 설정한다. 옹근수 및 류점수명령인 경우 그 개별적인 대기렬들은 FIFO 대기렬이라기보다 예약부로서의 기능을 수행한다. 즉 명령은 예약비트에 의하여 표시된데 따라 자료의존성에 따르는 임의의 순서로 출력될수 있다. 명령원천연산수에 대한 예약비트가 모두 동작상태가 아닌 경우 명령은 그의 실행장치가 준비되자

마자 통과된다. 주소대기렬은 읽기, 쓰기명령을 위한것으로서 순서출력규약을 리용하였다.

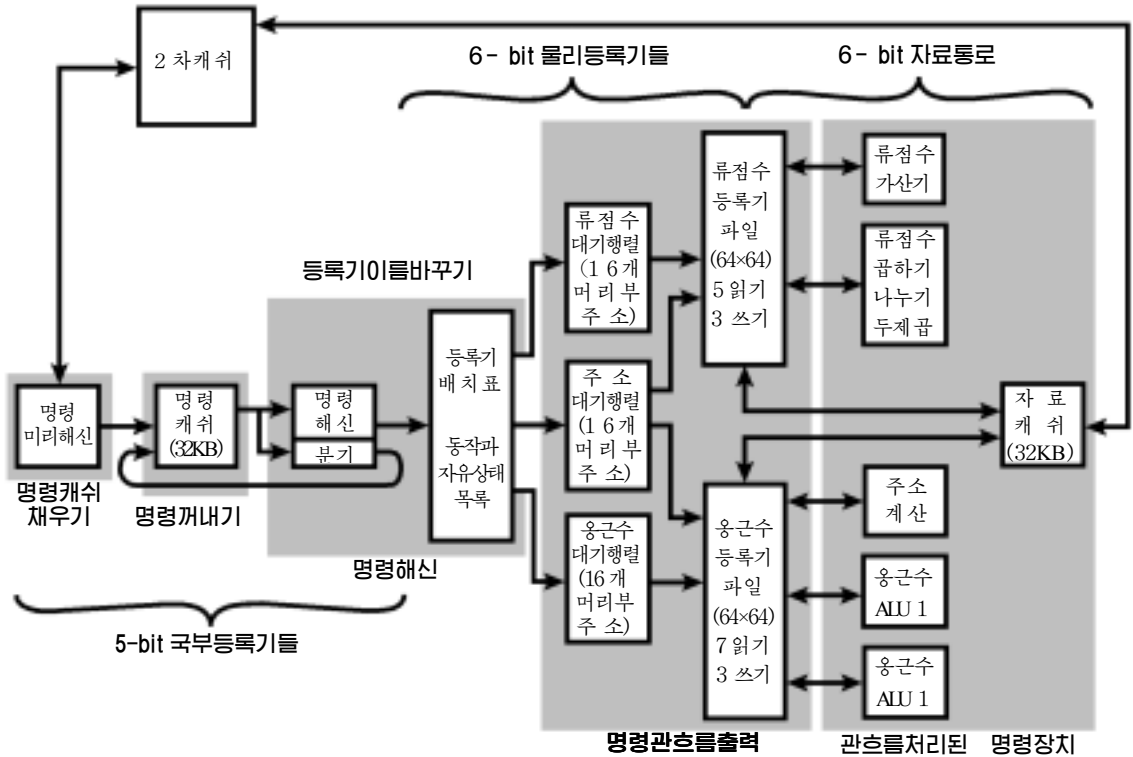



그림 13-15. MIPS 10000의 구조

옹근수와 류점수등록기파일들은 각각 64개의 물리등록기로 되여 있다. 실행장치는 등록기파일로부터 직접연산수를 읽어 그 결과를 다시 직접 쓰기한다. 다중읽기/쓰기포구들은 매개 파일에서 병렬호출을 허락할수 있다.

R10000에는 5개의 실행장치가 있는데 그것은 주소계수기, 두개의 옹근수산수-론리연산장치, 류점수더하기장치, 나누기와 두제곱뿌리계산의 다중화를 위한 류점수장치로 되여 있다. 두개의 옹근수 산수-론리연산장치는 더하기, 덜기, 론리연산을 할수 있다. ALU1은 또한 밀기와 분기조건조작을 진행한다. ALU2는 옹근수곱하기와 나누기연산을 진행한다.

제 6 절. UltraSPARC Ⅱ

UltarSPARC Ⅱ는 SPARC 처리장치로부터 나온 슈퍼스칼라기계이다. 이 처리장치는 제 12장 제 7절에서 서술한것처럼 많은 등록기와 등록기창문을 리용하는것을 비롯하여 동일한 명령모임구성과 같은 RISC 특성을 공유하고 있다.

1. 내부구성

그림 13-16 에 Uitra SPARC Ⅱ라고 부르는 최근에 나온 처리장치의 내부구성을 보여주었다. 내부의 L2 캐쉬는 따로 있는 L2 캐쉬에 자료와 명령을 준다. 미리꺼내기 및 선택장치(PPU)는 12 개의 명령까지 보관할수 있는 명령완충기에서 명령을 꺼낸다. PPU 는 2bit 리력방식을 가진 분기리력표를 리용하여 분기예측을 한다.

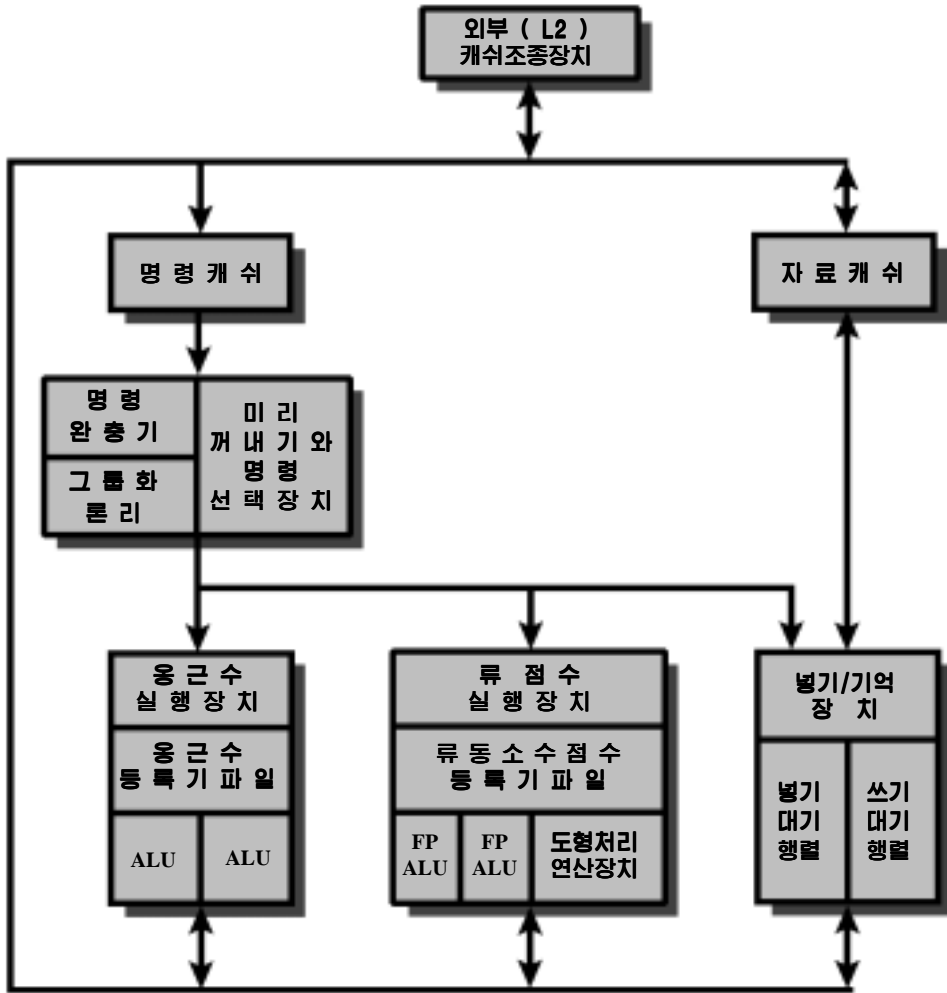



그림 13-16. UltraSPARC-Ⅱ의 블로크도

마지막으로 PPU 는 그룹으로 된 론리모듈도 가지고 있다. 그것은 선택을 동시적으로 하기 위하여 4 개 명령까지 그룹으로 묶어서 들어 오는 명령을 조직하기 위해서이다. 매 그룹에는 2 개의 옹근수명령과 2 개의 류점수명령/도형처리명령이 있어야 한다.

옹근수실행장치는 2 개의 ALU 와 8 개의 등록기창문으로 구성되여 있다. 또한 이 장치에서도 두개의 옹근수명령을 병렬로 실행할수 있다. 류점수장치는 두개의 류점수 ALU 와 도형처리장치로 되여 있다. 이것은 두개의 류점수명령이나 하나의 류점수, 하나의 도형처

리명령을 병렬로 실행할수 있다. 도형처리장치는 SPARC 명령모임에 비쥬얼명령(VIS)들을 지원하고 있다. Pentium Ⅱ의 MMX 명령과 같이 VIS 는 도형처리를 위하여 특별히 설정한 명령이다.

읽기/쓰기장치(LSU)는 모든 읽기 및 쓰기를 위한 주소를 만들어 내며 한주기에 하나의 읽기 또는 쓰기를 하게 한다. 그러나 넣기대기렬과 결합하여 하나의 L2 캐쉬를 호출할 때 같은 8byte 주소범위까지 다중기억을 압축할수 있다.

2. 관흐름

UitraSPARC Ⅱ는 옹근수와 류점수 및 도형처리명령을 위한 두 부분으로 구성된 9 단의 명령관흐름(그림 13-17)을 리용한다. 옹근수명령을 위하여 다음과 같은 관흐름단이 리용된다.

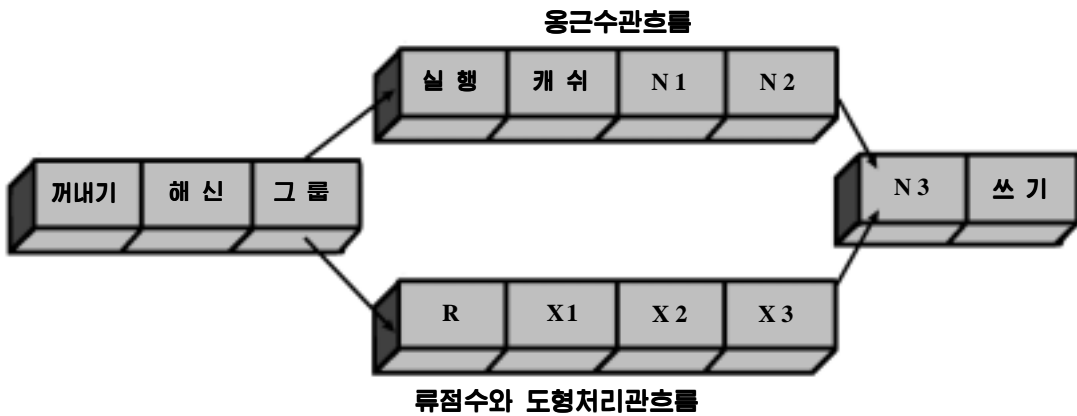



그림 13-17. UltraSPARC-II 의 명령관흐름

- **꺼내기단:** 한번에 4 개까지의 명령을 명령캐쉬로부터 꺼낸다.
- **해신기단:** 이 단은 명령을 해신하여 그것을 명령완충기에 넣는다.
- **그룹단:** 이 단은 명령완충기로부터 4개까지 명령을 선택하여 동시꺼내기를 한다. 이 그룹단에 의하여 두개의 옹근수명령이름과 두개의 류점수 및 도형처리명령이 선택될수 있다.
- **실행단:** 두개의 옹근수 ALU 는 옹근수등록기파일을 호출하여 옹근수명령을 실행한다. 이 단은 또한 명령을 읽기/쓰기하는데 필요한 가상주소를 계산한다.
- **캐쉬읽기단:** 이 단은 앞단에서 만들어 진 가상주소가 캐쉬와 일치하는가, 일치하지 않는가 하는것을 결정할수 있다. 실행단으로부터 제기된 ALU 는 이 단에서 조건코드를 발생시키며 이것을 미리꺼내기 및 선택장치에 보내여 분기예측을 검사한다. 만일 예측오유가 있으면 관흐름안에 이미 있던 명령은 지워 진다.
- **N1 단:** 만일 캐쉬에 대한 불일치가 생기면 L2 캐쉬와 주기억기에 대한 호출을 위하여 명령을 읽기/쓰기대기렬에 넣는다.
- **N2 단:** 이단은 류점수연산관흐름을 위한 대기단이다.
- **N3 단:** 트레프는 이 단에 있는 기간 해결된다.
- **Write 단:** 모든 결과는 이 단에서 등록기파일로 써넣어 진다.

류점수명령관흐름은 옹근수관흐름의 앞의 세단과 뒤의 두단을 공유하고 있다. 류점관흐름의 고유한 단은 아래의 4개단이다.

- **R 단:** 이단에 있는 기간 류점수과 도형처리명령은 다시 해신되며 등록기파일이 호출된다.
- **X1 단:** 류점수명령과 도형처리명령은 이 단에 있는 기간 자기의 실행을 시작한다.
- **X2 단:**실행은 이 단에 있는 기간 완료된다.

제 7 절. IA-64/Merced

8086 으로부터 시작되였으며 가장 성공적인 콤퓨터제품이였던 극소형처리장치계렬인 Pentium Ⅱ와 Pentium Ⅲ이 자기의 존재를 끝마치고 있다. Intel은 Hewlett-Packard(HP)와 팀을 묶어 IA-64라고 불리우는 새로운 64bit기본방식을 개발하고 있다. IA-64는 Intel회사의 32bit x86방식을 64 bit로 확장한것은 아니며 Hewlett-Packard의 64bit PA-RISC구성방식에 맞춘것도 아니다. IA-64 는 두 회사와 대학들에서 여러해동안 연구하여 만든 새로운 구성방식으로 되여 있다. 이 방식은 병렬화를 체계적으로 리용함으로써 새로운 새 세대의 마이크로소편에 방대한 회로와 높은 속도를 실현할수 있게 하였다.

IA-64의 기초로 되는 기본개념은 다음과 같다:

- 처리장치에 의하여 실행시간이 결정되는것이 아니라 기계명령에 의한 완전한 전용명령병렬화
- 긴 명령단어나 대단히 긴 명령단어(LIW/VLIW)
- 분기예측(분기예측과 다름)
- 투기적인 읽기

Intel 과 Hewlett-Packard 는 이 개념들의 조합을 완전한 전용병렬명령처리(EPIC)라고 한다. 이 회사들은 EPIC 라는 말을 기술 혹은 기술의 집합체라는 의미로 리용하고 있다. IA-64는 EPIC기술을 리용하여 실현한 실제적인 명령모임이다. 이 IA-64에 기초하여 처음으로 개발하려고 계획한 제품은 메르쎄드(Merced) 코드이다. 같은 IA-64 구성방식을 가진 다른 제품들이 계속 나오고 있다.

표 13-2 는 IA-64 와 전통적인 슈퍼스칼라방식사이의 주요한 차이점을 종합한것이다. 이 차이점에 대하여서는 이 절의 마지막에서 론의한다.

1. 필요성

Intel 은 x86 명령방식과 장치적호환성이 없는 새로운 기본방식에로 전환하는 문제를 순간적으로 결정하였지만 이것은 기술발전의 영향을 받게 되였다. 1970년대 말에 x86계렬이 개발되기 시작했을 때 처리장치에는 3극소자가 수만개정도 집적화되였지만 슈퍼스칼라장치로서는 낮은 수준이였다. 따라서 명령은 한번에 하나씩 실행되였으며 관흐름기술은 거의 리용되지 않았다. 1980년대 중엽에 3극소자의 수가 수십만개로 늘어 나면서 Intel은 관흐름기술을 도입하기 시작하였다(그림 11-18). 한편 다른 제작자들도 3 극소자의 수를 늘이고 RISC 방법에 기초하여 속도를 증가시키려고 시도하였다. 그리하여 관흐름의 효과를 높이기 위한 대책이 세워 졌으며 그후 다중실행장치를 실현할수 있는 슈퍼스칼라와 RISC가 결합된 기술이 도입되였다.

표 13-2. 전통적인 슈퍼스칼라 대 IA-64 기본방식

슈퍼스칼라	IA-64
단어당 하나의 RISC-행명령	세개의 그룹으로 된 RISC-행명령
다중병렬실행장치	다중병렬실행장치
실행시간에 명령흐름의 재순서화 및 최적화	번역시간에 명령흐름의 재순서화 및 최적화
한경로의 추측실행을 가진 분기예측	한갈래의 두경로에서 추측실행
필요할 때에만 기억기로부터 자료를 넣기, 캐쉬전반에 있는 자료의 탐색시도	그것이 필요되기전에 자료를 추측하여 넣기, 캐쉬전반에 있는 자료의 탐색시도

Pentium 을 개발하면서 Intel 은 두개의 CISC 명령을 동시에 집행하기 위하여 슈퍼스칼라기술을 도입하기 시작하였다. 또한 Pentium Pro 와 Pentium Ⅱ가 CISC 명령으로부터 RISC 류형의 마이크로연산과 더 높은 슈퍼스칼라기술을 리용할수 있게 하였다. 그러나 Pentium Ⅱ와 Ⅲ 이후세대의 처리장치에서는 한개 소편에 수천만개의 3 극소자를 쓰게 되였다.

처리장치설계자들은 이 문제를 해결하기 위한 몇가지 방안을 제기하였다. 한가지 방안은 이 과잉되는 3 극소자들을 소편우의 캐쉬에 넣는것이다. 캐쉬가 커지면 실리가 적어 지지만 성능은 개선할수 있으며 캐쉬를 더 크게 하면 명중률이 얼마간 개선된다. 다른 한가지 방안은 실행장치를 더 늘이는 방법으로 슈퍼스칼라화하는것이다. 이렇게 하면 설계가 매우 복잡해 진다. 실행장치를 더 늘이면 처리장치가 커지게 되며 실행장치들사이 동기를 보장하기 위한 론리회로가 보다 복잡해 진다. 따라서 분기예측을 개선하고 비순서처리를 넣어야 하며 관흐름을 길게 하여야 한다. 그런데 관흐름을 더 많이, 더 길게 하면 예측이 틀려 질 가능성이 더 커진다. 비순서실행을 도입하려면 더 많은 개명등록기들과 종속성을 관리하기 위한 복잡한 내부조종회로가 요구된다. 결과적으로 오늘날 가장 훌륭한 처리장치들은 최대로 한주기에 4 개 명령을 출력시킬수는 있으나 일반적으로는 그보다 적다.

이 문제를 해결하기 위하여 Intel 과 HP 는 많은 병렬실행장치를 가진 처리장치들을 효과적으로 리용할수 있는 전반적인 설계방안을 제안하였다. 이 새로운 방안에서 기본은 완전한 병렬화의 개념이다. 이 방안에서는 처리장치가 실행시간내에 명령을 동적으로 순서화하는것이 아니라 콤파일러가 번역창문시간내에 명령을 정적으로 순서화한다. 콤파일러는 명령들을 병렬로 실행할수 있으며 기계어명령과 관련된 이 정보를 포함하고 있다는것을 결정한다. 처리장치는 이 정보를 리용하여 병렬실행을 한다. 이러한 방안의 우점은 EPIC 처리장치가 비순서슈퍼스칼라처리장치처럼 회로가 복잡하지 않다는것이다. 더우기 처리장치가 잠재적인 병렬실행순간을 결정하는데 수나노초 걸리지만 콤파일러는 오히려 코드를 시험하고 프로그람을 전체적으로 검사하는데 훨씬 더 많은 시간이 걸린다.

2. 구성

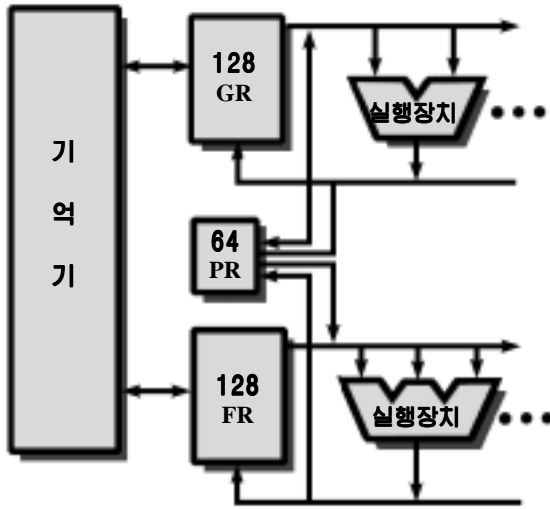
IA-64 는 다른 처리장치에서와 마찬가지로 여러가지로 구성할수 있다. 그림 13-18 에 IA-64 기계의 구성을 보여 주었다. 주요특징은 다음과 같다.

GR - 일반옹근수등록기
FR - 류동소수점수 및 도형처리등록기
PR - 한비트예측등록기
EU - 실행장치

그림 13-18. IA-64 방식의 일반구성

- 등록기가 많은것: IA-64 명령형식은 옹근수, 론리 그리고 일반용으로 128 개 등록기, 류점수와 도형처리용으로 128 개 등록기, 총 256 개의 64bit 등록기를 리용할수 있다.
- 여러개의 실행장치를 가진것: 오늘날 전형적으로 상품화된 슈퍼스칼라기계는 처리장치의 옹근수와 류점수 두 부분안의 4 개의 병렬실행장치를 리용하여 4 개 병렬관흐름을 지원한다. IA-64 는 앞으로 8 개 혹은 그이상의 병렬장치를 가진 체계로 될것이다.

IA-64 의 등록기파일은 대부분의 RISC 와 슈퍼스칼라기계에 비하면 매우 많다. 그 리유는 높은 수준의 병렬화를 지원하자면 많은 수의 등록기가 요구되기때문이다. 전통적인 슈퍼스칼라기계에서 기계어(아쎔블리언어)는 적은 수의 등록기를 리용하며 처리장치는 등록기개명기술과 의존성해석을 리용하여 이것을 보다 많은 수의 등록기들에 배치한다.

실행장치의 수는 특별히 제작한 3 극소자의 수에 관계된다. 처리장치는 그것이 가능한 범위까지 병렬화를 해야 한다. 례하면 만일 기계어명령의 흐름이 8 개의 옹근수명령을 병렬로 집행한다면 4 개의 옹근수관흐름을 가진 처리장치는 이것을 두 부분에서 실행하게 된다. 8 개의 관흐름을 가진 처리장치는 8 개의 명령을 동시에 다 실행할수 있다.

3. 명령형식

IA-64 는 길이가 128bit 묶음으로 정의되여 있는데 이것은 3 개의 명령과 템플레트마당으로 구성되여 있다(그림 13-19). 처리장치는 명령을 한번에 한묶음씩 꺼낼수 있으며 매 꺼내기마다 3 개 명령을 호출한다. 템플레트마당에는 명령을 병렬로 집행할수 있다는것을 지적하는 정보가 들어 있다. 템플레트마당에 대한 해석은 하나의 묶음에 국한된것이 아니다(하나의 묶음에 의하여 결정되지 않는다.). 오히려 처리장치는 여러개의 묶음을 검사하여 명령을 병렬로 실행할수 있는가를 결정한다. 례하면 명령흐름은 8 개 명령을 병렬로 실행할수 있으나 콤파일러는 이 8 개의 명령의 린접묶음을 련결할수 있도록 명령의 순서를 재배치하며 처리장치가 이 8 개의 명령이 독립이라는것을 알수 있도록 템플레트비트를 설정한다.

묶어 진 명령들은 본래의 프로그람순위와 다르다. 더우기 템플레트마당의 유연성으로 하여 콤파일러는 같은 묶음안에 독립명령과 종속명령을 함께 넣을수 있다. 선행한 VLIW 설계들과는 달리 IA-64 는 묶음을 채우기 위하여 무기능연산(NOP)을 넣을 필요가 없다.

PR - 예측등록기
GPR - 일반등록기(옹근수 혹은 류점수등록기)

그림 13-19. IA-64의 명령형식

매개 명령은 40bit의 고정길이형식으로 되여 있다. 이것은 RISC와 RISC슈퍼스칼라기계에서 찾아 볼수 있는 전통적인 32bit 코드보다(비록 그것이 Pentium II의 118bit 짜리 마이크로조작코드보다 훨씬 짧기는 하지만) 훨씬 더 길다. 첫째로 IA-64는 전형적인 RISC기계보다 훨씬 더 많은 등록기 즉 128개의 옹근수등록기와 128개의 류점수등록기를 가지고 있다. 둘째로 예측실행기술을 도모하기 위하여 IA-64 기계는 64개의 예측등록기를 가지고 있다. 예측등록기리용은 프로그람작성자와는 무관계하지만 예측등록기참조는 콤파일러에 의하여 명령안에 규정되게 된다. 그의 리용을 아래에서 설명한다.

4. 예측실행

IA-64의 주요한 두가지 기술은 예측실행과 투기넣기기술이다. 그림 13-20은 이 두 기술에 대하여 보여 주었는데 이에 대해서는 이 소제목과 다음소제목에서 론의하게 된다.

예측은 콤파일러가 어느 명령을 병렬로 실행할수 있는가를 결정하는 기술이다. 번역과정에 콤파일러는 조건실행을 리용하여 프로그람에서 분기를 제거한다. 고수준언어에 의한 전형적인 실례는 **if-then-else** 명령이다. 전통적인 콤파일러는 이 구문의 **if** 점에 조건분기를 넣는다. 만일 조건이 하나의 론리적결과를 가진다면 분기를 취할수 없으며 **then**경로에 대응하는 다음의 명령블로크가 실행된다. 그리고 이 경로의 끝에 **else**경로에 따르는 다른 블로크에 대한 무조건분기가 있다. 만일 조건이 다른 론리적출력를 가진다면 분기는 명령의 **then**블로크에서 취해 지며 명령의 **else**블로크에서 실행을 계속한다. 두개의 명령흐름은 **else** 블로크가 끝난 다음 서로 련결된다. IA-64 콤파일러는 다음과 같은것을 한다(그림 13-20 ㄱ).

1. 프로그람의 **if** 점에 비교명령을 넣어 두개의 예측을 만든다. 만일 비교가 진리이면 첫번째 예측은 진리로 설정되며 두번째것은 거짓으로 설정된다. 그리고 비교가 거짓이면 첫번째 예측은 거짓으로 설정되며 두번째것은 진리로 설정된다.
2. 첫번째 예측의 값을 가지고 있는 예측등록기를 참조하여 **then** 경로안의 매개 명령을 늘이며 두번째 예측의 값을 가지고 있는 예측등록기를 참조하여 **else** 경로안에

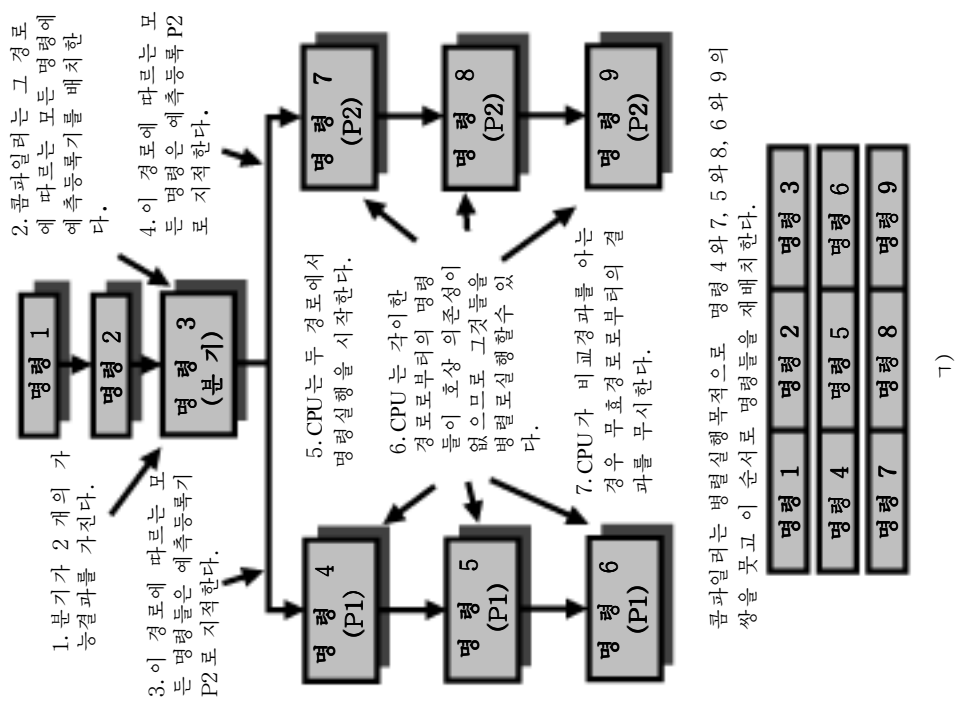
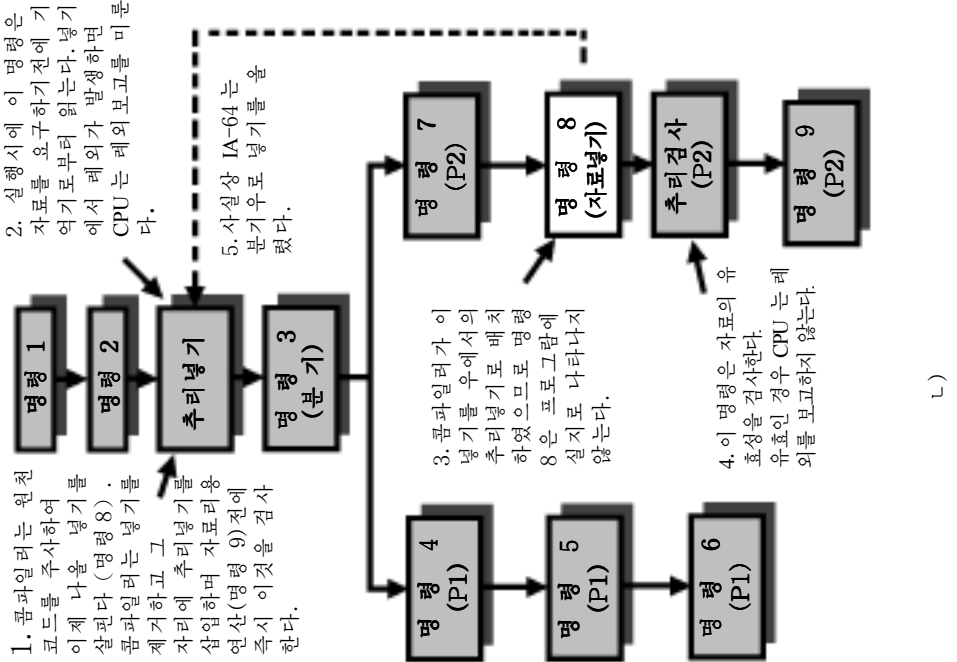



그림 13-20. IA-64 예측과 추리넣기
ㄱ-예측 , ㄴ- 추리넣기

있는 매개 명령을 늘인다.
3. 처리장치는 두 경로에 따라서 명령을 실행한다. 비교결과가 결정되면 처리장치는 어느 한경로에 따르는 결과를 버리고 다른 경로에 따라 결과를 기억해 둔다.

실례로 다음의 원천코드를 보기로 하자

원천코드:

```
if (a && b)
   j = j + 1 ;
else
    if (c)
         k = k + 1 ;
    else
         k = k - 1 ;
    I = I + 1 ;
```

두개의 if명령문은 3개의 가능한 실행경로가운데서 어느 하나를 선택한다. 이것은 다음의 아쎔블리언어로 번역되는데 3개의 조건분기와 하나의 무조건분기명령으로 되여 있다.

아쎔블리언어코드:

```
        Beg   a,0,L1
        Beg   b,0,L2
        Jump  L3
L1:     beg   c,o,L2
        Add  k,k,1
        jump  L3
L2:     sub  k,k,1
L3:     add  I,I,1
```

그림 13-21은 이 아쎔블리언어코드의 흐름도를 보여 준다. 이 흐름도는 아쎔블리언어프로그람을 여러개의 코드블로크로 나누어 표시하였다. 조건에 따라서 집행하는 매개 블로크에 대하여 콤파일러는 예측을 할당할수 있다. 이 예측들을 그림 13-21에 보여 주었다. 이 예측이 모두 거짓으로 초기화되였다고 하면 그 결과코드는 다음과 같다.

예측된 코드 :

```
(1)  p1, p2 = cmp (a == 0)
(2)  <p2> p1, p3 = cmp ( b== 0)
(3)  <p3> add j, j, 1 결과코드
(4)  <p1> p4, p5 = cmp (c != 0)
(5)  <p4> add k, k, 1
(6)  <p5> sub k, k, 1
(7)  add i, i, 1
```

여기에 3개의 구문이 있다.

<PI>   명령

이러한 형의 명령은 예측 PI가 진리인 경우만 집행되는 형이다. 그림 13-19의 명령형식의 견지에서 PR마당은 1bit의 예측등록기 I를 지적하기 위하여 값 I를 가진다. 처리장치는 이 명령을 꺼내여 해신한 다음 실행을 시작하지만 예측등록기 I의 값을 1(진리) 혹은 0(거짓)으로 정한 다음 결과를 기억하겠는가 안하겠는가를 결정한다.

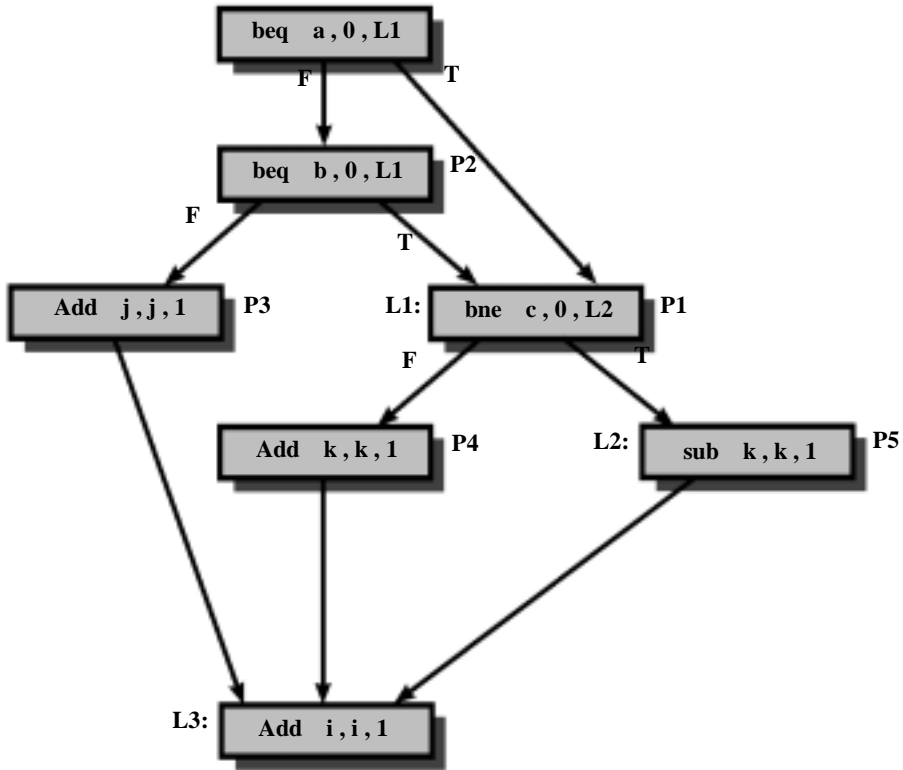



그림 13-21. 예측의 실례

다음과 같은 형태의 명령

PJ, PK = cmp (relation)

은 그 관계가 진리이면 예측값 J 를 진리로, K 의 값을 거짓으로 설정하며 관계가 거짓이면 예측값 J 를 거짓으로, K 를 진리로 설정하게 된다. 이 명령형식은 예측등록기-등록기명령의 형식을 보여 준 그림 13-19 와는 다르다. 좀더 구체적으로 말하면 이 명령은 두개의 예측등록기마당을 가지고 있다. 처리장치는 이 명령을 꺼내여 해신한 다음 실행하여 그 결과를 예측등록기들인 J 와 K 에 넣는다.

마지막으로 예측발생명령은 자체로 예측할수 있다.

<PI> PJ, PK = cmp (relation)

예측등록기 J 와 K 는 예측 PI 가 진리인 경우 비교에 기초하여 갱신된다. 만일 예측 PI 가 거짓이면 그때 명령은 실행되지 않는다. 이 명령은 자기의 형식안에 3 개의 예측등록기마당을 요구한다.

프로그람을 다시 살펴 보면 아쎔블리언어코드안의 첫 두개의 조건분기명령은 두개의 예측비교명령으로 번역된다. 만일 첫번째 명령이 P2 를 거짓으로 설정하면 두번째 명령은 집행되지 않는다. 원천코드의 바깥쪽 if 명령문이 진리이면 P3 은 진리로 된다. 바깥쪽 if 명령문의 then 부분이 이러한 리유로 P3 에 의하여 예측된다. 예측코드의 명령(4)은 바깥

쪽 else 부분의 더하기 혹은 덜기명령을 수행하겠는가 안하겠는가를 결정한다. 마지막으로 I 를 무조건 증가시킨다. 원천코드를 고찰한 다음 예측코드를 들여다보면 명령(3), (5), (6)가운데서 어느 하나만 실행된다는것을 알수 있다. 흔히 쓰는 슈퍼스칼라처리장치에서 분기예측을 리용하여 이 셋중의 어느 하나를 집행하며 그 경로를 따라 가겠는가를 예측한다. 처리장치가 오유를 발견하면 관흐름은 재설정된다. IA-64 처리장치는 이 세개의 명령을 다 실행할수 있으며 일단 예측등록기의 값이 정해 지면 유효명령의 결과만을 기억한다. 따라서 보충된 병렬실행장치들을 리용하면 관흐름재설정으로 인한 지연을 방지한다.

예측실행에 대한 초기연구들은 대부분 일리노이스(Illinois)대학에서 진행되였다. 연구자들이 진행한 모의연구는 예측을 리용하는것이 다중병렬관흐름처리장치인 경우 실제적으로 동적분기 및 분기예측오유가 감소되고 성능이 개선된다는것을 밝혔다 [MAHL94], [MAHL95].

5. 투기넣기

IA-64 가 거둔 또 하나의 성과는 투기넣기기술이다. 이것은 처리장치로 하여금 기억호출지연을 피하기 위하여 프로그람이 요구하기전에 자료를 기억기로부터 미리 꺼내는 기술이다. 또한 처리장치는 례외를 통지하는것이 필요할 때까지 례외통지를 연기한다. 용어 **끌어올리기**는 명령흐름의 앞쪽으로 넣기명령을 옮기는것을 말한다.

넣기지연을 최소화하는것은 성능개선에서 중요한 자리를 차지한다. 일반적으로 이미 코드블로크안에는 기억기로부터 등록기에로 자료를 전송하는 여러개의 넣기명령이 있다. 왜냐하면 한준위 혹은 두준위의 캐쉬를 더 넣는다고 하여도 기억기는 슈퍼스칼라처리장치에 비하면 속도가 뜨기때문에 기억기로부터 자료를 넣기할 때 호출지연이 문제로 된다. 이것을 최소로 줄여 될수록 고속으로 넣기 위하여 코드를 재배렬하군 한다. 이것은 임의의 콤파일러를 가지고 할수 있다. 이러한 경우는 조종흐름을 따라서 넣기를 하는 경우이다. 넣기는 분기보다 앞에서 무조건넣기를 할수 없는데 그 리유는 넣기가 실지로 생기지 않을수도 있기때문이다. 자료를 기억기로부터 넣을수 있지만 예측결과가 알려 질 때까지 방식등록기에 기억되지 않도록 예측을 리용하여 조건에 따라 넣기를 할수 있다. 이 방식이 가지고 있는 문제점은 넣기가 많을수 있다는것이다. 무효주소나 페지오유에 의한 례외가 생길수 있다. 이것이 생기면 처리장치는 지연을 발생시키는 요인으로 되는 례외나 오유를 처리하여야 한다.

그러면 분기보다 먼저 어떻게 넣기를 할수 있는가? IA-64 에서는 이 문제를 투기넣기기술로 해결하였다. 여기서는 례외실행으로부터 넣기실행을 분리한다(그림 13-20 ㄴ). 원천프로그람에 있는 넣기명령은 두개의 명령으로 교체한다.

- 투기넣기(ld.s)는 기억호출을 진행하여 례외검사를 하지만 례외를 처리하지는 않는다(OS 루틴을 호출하여 례외를 조종한다). ld.s 명령은 프로그람안에서 선행한 적당한 곳으로 올리 밀게 된다.
- 검사명령(cheek.s)은 본래의 넣기위치에 있으면서 례외를 관리한다. 이 cheek.s명령은 예측이 진리인 경우만 실행할수 있도록 예측된다.

만일 ld.s 가 례외를 검사하면 목표등록기와 관련된 일어남(token)비트를 설정한다. 대응하는 cheek.s 명령이 실행되고 일어남비트가 설정되여 있으면 cheek.s 명령은 례외조종루틴에로 이행한다.

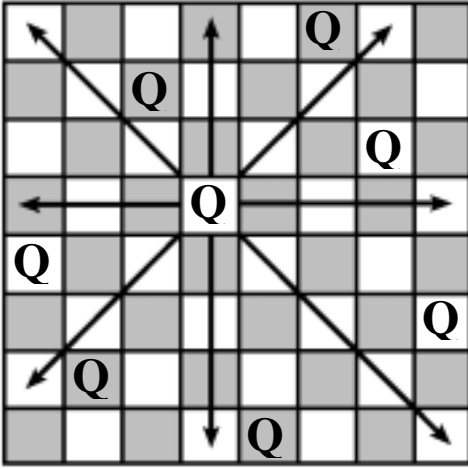



그림 13-22. 여덟왕에 대한 문제

예측프로그람의 성능을 검사하며 투기넣기의 리용을 증명하기 위하여 Intel 회사와 HP 회사가 리용한 실례는 《여덟명의 왕》문제이다. 풀이는 어느 한 왕이 임의의 다른 왕을 위협하지 않도록 여덟명의 왕을 장기판우에 배렬하는것이다. 그림 13-22에 그 한가지 풀이를 보여 주었다. 내부순환에서 원천코드의 기본행은 다음과 같다

```
if ((b[j] == true && (a[i +j] == true) && (c[i -j + 7] == true))
```

총적과정은 렬이나 두개의 대각선에 이미 배치된 왕을 다치지 않게 왕을 매개 행에 놓으면서 행을 따라 움직이는것이다. B배렬은 렬을 검사하며 A와 C배렬은 두개의 대각선을 검사한다.

아쎔블리프로그람은 3개의 넣기명령과 3개의 분기명령으로 되여 있다.

아쎔블리언어코드:

```
(1)          R1 = &b[j]
(2)          Ld R2, (R1)
(3)          Bne R2, 1, L2
(4)          R3 = &a[i + j]
(5)          ld R4, (R3)
(6)          Bne R4, 1, L2
(7)          R5 = &c[i - j + 7]
(8)          ld R6, (R5)
(9)          bne R6, 1, L2
(10)  L1:    <code for then path>
(11)  L2:    <code else path>
```

투기넣기와 예측실행을 리용하면 코드는 다음과 같이 된다.

투기와 예측코드:

```
(1)          R1 = &b[j]
(2)          R3 = &a[i + j]
(3)          R5 = &c[i - j + 7]
(4)          ld R2, (R1)
(5)          ld.s R4, (R3)
(6)          ld.s R6, (R5)
(7)          P1, P2 = cmp (R2 == 1)
(8)          <P2> br L2
(9)          chk.s R4
(10)         P3, P4 = cmp (R4 == 1)
(11)         <P4> br L2
(12)         chk.s R6
(13)         P5, P6 = cmp (R5 == 1)
(14)         <P6> br L2
(15)  L1:    <code for then path>
(16)  L2:    <code for else path>
```

아쎔블리언어프로그람은 3 개의 기본코드블로크로 되여 있는데 그의 개개는 조건분리를 가진 넣기로 되여 있다. 아쎔블리언어코드에서 주소설정명령 4와 7은 간단한 산수연산명령이다. 이것들은 임의의 시각에 실행될수 있으므로 콤파일러는 이것을 제일 우에까지 옮길수 있다. 이때 콤파일러는 3 개의 블로크와 맞다 들게 된다. 이 블로크 개개는 넣기, 조건계산, 조건분기로 구성되여 있다. 여기에서는 얼마간 병렬로 실행될수 있다. 더우기 넣기가 둘 혹은 그이상의 박자주기가 요구된다고 가정하면 조건분기가 실행되기전에 얼마간의 시간이 랑비된다. 콤파일러가 해야 할것은 모든 분기의 우에 두번째와 세번째넣기(아쎔블리언어코드에서 명령 5 와 8)를 올려 보내는것이다. 이것은 투기넣기를 제일 우에(명령 5와 6) 놓고 원천코드블로크(명령 9와 12)에서 검사를 하는 방법으로 진행한다.

이 변환은 3 개의 넣기를 모두 병렬로 실행할수 있게 하며 넣기도달시간으로 하여 생기는 지연을 최소로 하든가 또는 피하기 위하여 미리넣기를 시작할수 있게 한다. 콤파일러는 예측리용을 더 심화시키면 3개의 분기가운데서 두개를 없앨수 있다.

투기 및 예측된 수정코드:

```
(1)          R1 = &b[j]
(2)          R3 = &a[i + j]
(3)          R5 = &c[i - j + 7]
(4)          ld R2, (R1)
(5)          ld.s R4, (R3)
(6)          ld.s R6, (R5)
(7)          P1, P2 = cmp (R2 == 1)
(8)          <P1> chk.s R4
(9)          <P1> P3, P4 = cmp (R4 == 1)
(10)         <P3> chk.s R6
(11)         <P3> P5, P6 = cmp (R5 == 1)
(12)         <P6> br L2
(13)   L1:   <code for then path>
(14)   L2:   <code for else path>
```

이미 두 예측을 발생시키는 한가지 비교를 진행하였다. 수정코드에서 콤파일러는 거짓예측에 대한 분기대신에 검사의 실행과 진리예측에 대한 다음비교를 한정한다. 두 분기를 제거한다는것은 두개의 잠재하고 있는 예측오유를 제거한다는것을 의미하며 이렇게 되면 두개이상의 명령이 절약된다.

참고문헌과 Web 싸이트

[JOHN91]은 슈퍼스칼라설계에 대한 내용을 취급한 좋은 책이다. 참고문헌[SIMT95]와 [SIMA97]은 이 문제에 대한 론문을 주로 취급하였다. [JOUP89a]는 명령준위병렬화에 대하여 보여 주고 병렬화를 최대로 실현하기 위한 여러가지 수법들을 고찰하였으며 모의방법으로 슈퍼스칼라와 슈퍼관흐름방법들을 비교하였다.

[POPE91]은 슈퍼스칼라기계를 보다 구체적으로 고찰하였다. 또한 비순차명령실행과 관련되는 설계항목들에 대한 훌륭한 교과서로 되고 있다. 제한된 체계에 대한 다른 각도에서의 고찰은 [KUGA91]에서 보여 주었다. 이 론문에서는 슈퍼스칼라실현에 대한 중요한 설계항목들을 고찰하였다. [LEE91]은 슈퍼스칼라성능을 개선하는데서 리용할수 있는 쏘프트웨어수법들을 서술하였다. [WALL91]은 명령준위병렬화를 슈퍼스칼라처리장치에서 리용하였다는것을 보여 주는 흥미 있는 연구자료를 주고 있다.

[SHAN98]은 Pentium Ⅱ에서의 명령관흐름을 매우 구체적으로 서술하였다. 보다 훌륭한 설계는 [PAPW96]에서 론의하였으며 여기에서는 Pentium Pro 를 위주로 하여 처리장치 구성설계에 미치는 영향의 일부를 해석하였다. Pentium Pro 는 Pentium Ⅱ, Ⅲ과 같은 슈퍼스칼라구성으로 되여 있지만 MMX 기능은 없다.

[POTT94]는 PowerPC 601 에서의 명령관흐름을 구체적으로 서술하였다. [SHAN95]도 좋은 참고서이다.

MIPS R10000 에 대한 구체적인 고찰은 [YEAG96]에서 서술하였다. UltraSPARC Ⅱ에 대한 론의는 [NORM89]에 서술되여 있다. [DULO94]는 IA-64 에 대해 개괄하였으며 [HWU98]은 예측실행에 대한 간단한 개요를 주었다.

DULO98 Dulong, C.《The IA-64 Architecture at Work.》*Computer*, July 1998.

HWU98 Hwu, W.《Introduction to Predicated Execution》, *Computer* , January 1998.

JOHN91 Johnson, M. *Superscalar Microprocessor Design*. Englewood Cliffs, NJ: Prwntice Hall, 1991.

JOUP89a Johnson, N., and Wall, D.《Available Instruction-Level Parallelism for Super- scalar and Superpipelined Machines.》*Proceedings*, *Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Apil 1989.

KUGA91 Kuga, M.; Murakami,K.; and Tomita, s. 《DSNS(Dynamically-Hazard Resolved, Statically-Code-Scheduled, Nonuniform Superscalar):Yet Another Superscalar Processor Architecture.》*Computer Architecture News*, June 1991.

LEE91 Lee, R.; Kwok, A.; and Briggs, F.《The Floating Point Performance of a Superscalar SPARC Processor.》*Proceedings*, *Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Apil 1989.

NORM98 Normoyle, K., et al.《UltraSPARC-Ⅱ i: Expanding the Boundaries of a Statem on a Chip.》*IEEE Micro*, March/April 1998.

PAPW96 Papworth, D.《Tuning the Pentium Pro Microarchitecture.》*IEEE Micro*, April 1996

POPE91 Popescu, V., et al.《The Metaflow Architecture.》*IEEE Micro*, June 1991.

POTT94 Potter, T., et al.《Resolution of Data and Control-Flow Dependencies in the PowerPC 601.》*IEEE Micro*, October 1994.

SHAN95 Shanley, T. PowerPC *System Architecture*. Reading, MA: Addison-Wesley, 1995.

SHAN98 Shanley, T. Pentium Pro and Pentium Ⅱ *System Architecture*. Reading, MA: Addison-Wesley, 1998.

SIMA97 Sima, D.《Superscalar Instruction Issue.》*IEEE Micro*, September/ October 1997.

SMIT95 Smith, J., and Sohi, G.《The Microarchitecture of Superscalar Processors.》*Proceedings of the IEEE*, December 1995.

WALL91 Wall, D.《Limits of Instruction-Level Parallelism.》*Proceedings*, *Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, Apil 1991.

YEAG96 Yeager, K.《The MIPS R 10000 Superscalar Microprocessor.》*IEEE Micro*, April 1996.

Web 싸이트:

- **Merced/IA-64**: IA-64 와 Merced 처리장치에 대한 최신정보를 가진 Intel 의 싸이트
- **IMPACT**: 이것은 Illinois 대학의 싸이트이다. 이 대학에서는 예측실행과 관련한 많은 연구를 진행하였다. 이 싸이트를 리용하면 이와 관련한 수많은 론문들을 볼수 있다.

련습문제

1. 비순차완료를 슈퍼스칼라처리장치에서 리용하면 새치기처리후 례외조건이 그 결과를 비순차적으로 만들어 진 명령처럼 검사할수 있으므로 실행을 다시 계속하는것은 복잡하다. 프로그람은 례외명령다음의 명령에서 다시 시작할수 없다. 그것은 다음명령이 이미 완료되였고 그렇게 하면 이 명령이 두번 실행되게 되는것으로 된다. 이러한 다룰수 있는 기구를 제기하시오.
2. 다음과 같은 순서로 되여 있는 명령을 고찰하여 보자. 여기서 매개 명령문은 하나 혹은 두개의 원천등록기와 목적등록기를 가진 조작코드로 이루어 져 있다.

0	ADD	R3, R1, R2
1	LOAD	R6, [R3]
2	AND	R7, R5, 3
3	ADD	R1, R6, R0
4	SRL	R7, R0, 8
5	OR	R2, R4, R7
6	SUB	R5, R3, R4
7	ADD	R0, R1, R10
8	LOAD	R6, [R5]
9	SUB	R2, R1, R6
10	AND	R3, R7, 15

4 개 단으로 된 관흐름 즉 꺼내기, 해신 및 관흐름출력, 실행, 되쓰기를 리용한다고 하면 모든 관흐름단은 실행단을 제외하고는 한박자주기를 취한다고 하자. 간단한 옹근수연산 및 론리명령을 실행하기 위하여 실행단은 한주기를 취하지만 기억기로부터 LOAD 때문에 실행단에서 5 주기가 소비된다.

만일 단일한 스칼라관흐름으로 되여 있으나 비순차실행을 할수 있다면 첫 6 개 명령의 실행을 위하여 다음과 같은 표를 구성할수 있다.

4 개 관흐름단에서의 머리부는 박자주기를 가리키는데 이것은 매개 명령에서의 매개 위상의 시작을 의미한다. 이 프로그람에서 두번째 ADD 명령(명령 3)은 그의

명 령	꺼내기	해 신	실 행	되쓰기
0	0	1	2	3
1	1	2	4	9
2	2	3	5	6
4	4	5	6	7
5	5	6	8	10
6	6	7	9	12

연산자가운데서 r6 에 대하여 LOAD 명령(명령 1)에 의존한다. 그것은 LOAD 명령이 5 박자주기를 취하기때문이다. 관흐름출력론리는 두박자후에 의존성을 가진 ADD 명령과 관계되므로 이것은 ADD 명령을 3 박자주기동안 지연하지 않으면 안된다. 비순차론리를 가짐으로써 처리장치들은 박자주기 4에서 명령 3을 지연할수 있으며 그때 다음 3개의 독립적인 명령이 관흐름출력된다. 그리고 그것은 박자 6, 8, 9 에서 실행에 들어 간다. LOAD 명령은 박자 9 에서 실행완료한다. 의존성 있는 ADD 는 박자 10 에서 실행에 들어 갈수 있다.

1) 처리표를 완성하시오.

2) 비순차가능성을 예측하는 표를 다시 작성하시오. 이 가능성을 리용함으로써 무엇을 절약할수 있는가?

3) 매개 단에서 한번에 두개 명령을 조종할수 있는 슈퍼스칼라실현을 예측하는 표를 작성하시오.

3. PowerPC 601 의 선택장치안의 명령대기렬에서 명령은 분기처리장치와 류점수장치에 비순차적으로 선택될수 있다. 그러나 옹근수연산장치에서 명령은 오직 대기렬의 아래로부터로만 선택될수 있다. 이 제한은 왜 있는가?

4. 다음의 경우에 대하여 그림 13-4 와 같은 그림을 그리시오.

1) 분기예측: 취한다: 정확한 예측: 분기가 취해 진다.

2) 분기예측: 취한다.: 부정확한 예측: 분기가 취해 지지 않는다.

5. 다음의 아쎔블리언어프로그람에 대하여 설명하시오

이 프로그람은 쓰기-쓰기, 읽기-쓰기, 쓰기-읽기의존성이 포함되여 있다. 이것을 찾으시오.

```
11:  Move   R3, R7       /R3 ← (R7)/
12:  Load   R8, (R3)     /R8 ← 기억기(R3)/
13:  Add    R3, R3, 4    /R3 ← (R3) + 4/
14:  Load   R9, (R3)     /R9 ← 기억기(R3)/
15:  BLE    R8, R9, L3   /(R9)＞(R8)이면 분기
```

6. 그림 13-23 에 슈퍼스칼라처리장치의 내부구성실례를 보여 주었다. 처리장치가 자원충돌이 없고 자료의존성문제가 없다면 한주기당 두개 명령을 출력할수 있다. 여기에는 주로 4 개의 처리단계(즉 꺼내기, 해신, 실행, 기억)를 가진 2 개의 관흐름

이 있다. 매개 관흐름은 자기자체의 꺼내기해신장치와 기억장치를 가지고 있다. 4개의 기능장치 (곱하기, 더하기,론리, 넣기장치)들은 실행단에서 리용할수 있으며 동적기지우의 두개의 관흐름에 의하여 공유된다. 두개의 기억장치는 두개의 관흐름에 의하여 동적으로 리용될수 있다. lookahead 창문은 자기자체의 꺼내기와 해신론리를 가지고 있다. 이 창문은 이 순차명령을 출력하기 위하여 명령을 lookahead 하는데 리용된다.

이 처리장치에서 실행되는 다음의 프로그람을 설명하시오.

11:	Load	R1, A	/R1 ← 기억기(A)/
12:	Add	R2, R1	/R2 ← (R2) + R(1)/
13:	Add	R3, R4	/R3 ← (R3) + R(4)/
14:	Mul	R4, R5	/R4 ← (R4) + R(5)/
15:	Comp	R6	/R6 ← (R6)/
16:	Mul	R6, R7	/R3 ← (R3) + R(4)/

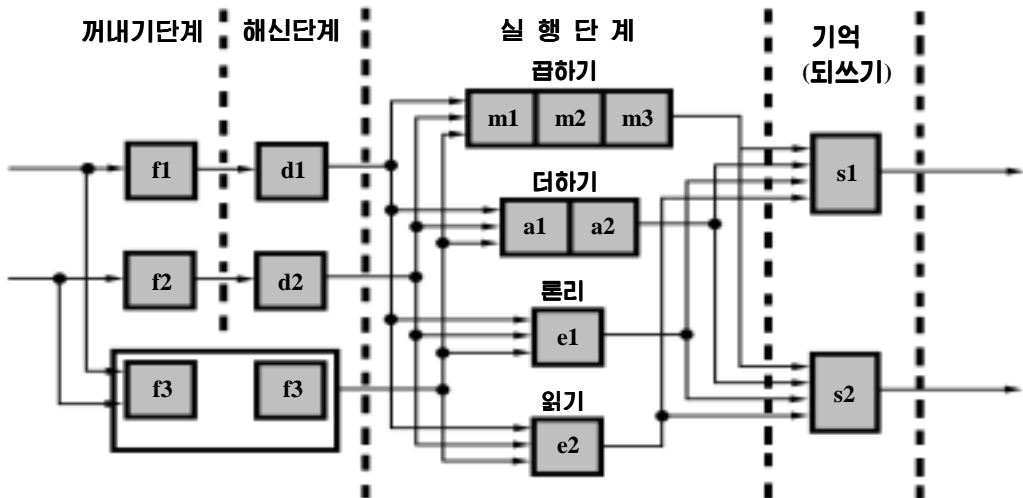



그림 13-23. 슈퍼스칼라처리소자의 2중관흐름처리

1) 프로그람안에 어떤 독립성이 존재하는가?

2) 순차완료방식으로 된 비순차출력과 그림 13-2 와 같은 표현을 리용한 그림 13-23 의 처리장치에 대한 우의 프로그람의 관흐름동작을 설명하시오.

3) 비순서완료를 가진 순서출력에 대하여 반복하시오.

4) 비순서완료를 가진 비순서출력를 반복하시오.

7. 그림 13-24 는 슈퍼스칼라설계에 대한 론문에 있는것이다. 그림의 3 부분을 설명하고 W, X, Y, Z 를 정의하시오.

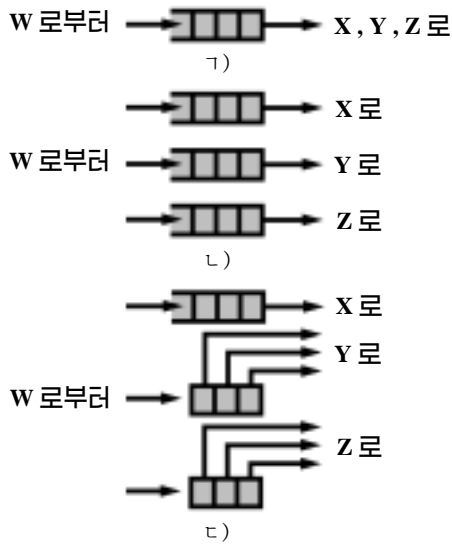



그림 13-24. 문제 7번의 그림

8. 제 13장 제 7절에서 예측실행을 위한 다음과 같은 구문을 도입하였다. 다음의 진리값표안에 값을 써넣어 완성하시오.

PI	비교	PJ	PK
없음	0		
없음	1		
0	0		
0	1		
1	0		
1	1		

9. 그림 13-21의 흐름도로 표현된 제 7절의 예측프로그람에서 다음의것을 지적하시오.

1) 병렬로 실행될수 있는 명령들을 지적하시오

2) 같은 IA-64명령모임에 속할수 있는 명령들을 지적하시오.

10. 아래의 원천프로그람토막을 고찰하고 다음의 물음에 대답하시오

```
for     (i = 0; i < 100; i ++)
        if  (A[i] < 50)
                j = j + 1;
        else
                k = k + 1;.
```

1) 대응하는 아쎔블리언어코드를 작성하시오.

2) 예측실행기술을 리용하여 결과를 아쎔블리언어코드토막으로 다시 작성하시오.

제 4 편. 조종장치

제 4 편의 중심

제3편에서는 매개 명령을 실행함에 있어서 처리장치에서 수행되는 기계어명령과 연산들에 중심을 두었다. 여기에서 론의하지 못한것은 매 개별적조작이 어떻게 일어 나는가 하는것이다. 이것을 담당하는것이 조종장치이다.

조종장치는 처리장치의 한 부분으로서 처리에 필요한 조종신호를 발생시킨다. 조종장치는 처리장치에 외부신호를 가하여 기억기 및 입출구모듈과의 자료교환을 진행하게 한다. 조종장치는 또한 처리장치에 내부조종신호를 보내여 등록기들사이에 자료를 이동하게 하며 ALU 로 하여금 규정된 기능을 수행하게 하거나 다른 내부조작을 조종하게 한다. 조종장치에 가해 지는 입구는 명령등록기, 기발, 외부원천(례하면 새치기신호들)으로 들어 오는 조종신호들이다.

제 4 편의 장별 내용

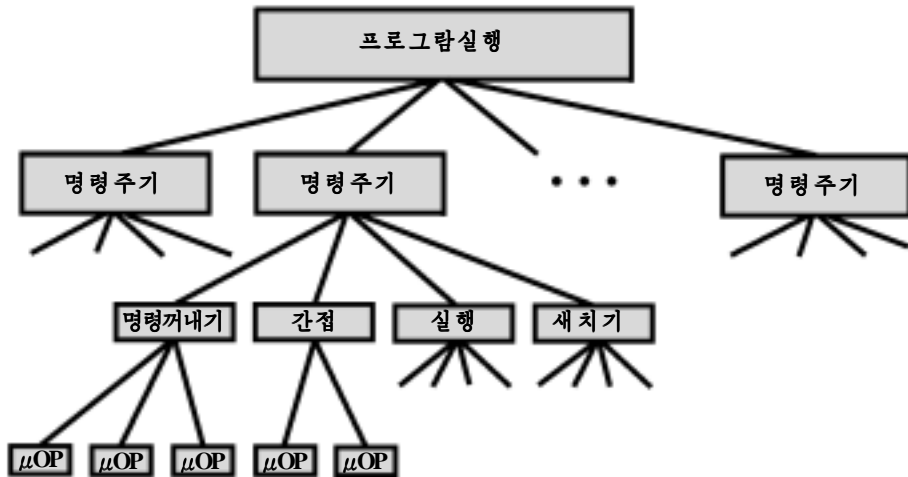
제 14 장. 조종장치조작

제 14 장은 조종장치가 수행해야 할 기능을 설명하면서 조종장치의 조작을 설명한다. 조종장치의 기본사명은 마이크로조작이라고 부르는 요소적인 조작순서를 명령주기기간에 발생시키는것이다.

제 15 장. 마이크로프로그람조종

제 15 장에서는 마이크로조작의 개념이 마이크로프로그람이라고 불리우는 조종장치실현의 강력한 방안으로 되였는가를 인식하게 된다. 본질은 저수준프로그람작성언어를 개발하는것이다. 처리장치의 기계어안의 매개 명령은 순차적인 저수준조종장치명령으로 번역된다. 이 저수준명령을 마이크로명령이라고 부르며 번역과정을 마이크로프로그람화라고 한다.

제 14 장. 조종장치조작

◆◆

◆하나의 명령실행은 일반적으로 주기라고 불리우는 여러가지 부분실행들의 모임으로 되여 있다. 례하면 하나의 명령실행은 꺼내기주기, 간접주기, 실행주기, 새치기주기로 구성되여 있다. 매개 주기는 차례로 마이크로조작이라고 불리우는 여러개의 기초적인 조작과정으로 되여 있다. 하나의 마이크로조작은 일반적으로 등록기사이 전송, 등록기와 외부모선사이전송 그리고 간단한 ALU 조작으로 구성되여 있다.

◆처리장치의 조종장치는 두가지 과제를 수행한다.

(1) 처리장치가 실행하려는 프로그람에 의하여 결정되는 마이크로조작을 일정한 순서로 실행하게 하며

(2) 매개 마이크로조작을 실행할수 있게 조종신호를 발생시킨다.

◆조종장치에 의하여 발생되는 조종신호는 론리문회로를 열기 및 닫기(ON, OFF)하여 얻어 지는데 그 결과 등록기에로 자료의 전송(읽기쓰기)과 ALU 조작을 한다.

◆조종장치를 실현하기 위한 한가지 기술은 배선론리식으로 장치를 실현하는것인데 여기서는 조종장치가 조합회로로 구성된다. 현재 기계명령에 의해 조종되는 그의 입구론리신호들은 한 묶음의 출구조종신호로 전송된다.

◆◆

제 9 장에서는 기계명령이 처리장치를 정의하는데 상당한 기일을 요구하였다는것을 지적하였다. 모든 조작코드의 효과와 주소지정방식을 포함한 기계어명령모임을 알고 있고 리용자가 볼수 있는 등록기묶음을 알고 있다면 처리장치가 수행해야 할 기능을 파악할수 있다. 이것은 완전한 파악을 의미하지 않는다. 이외에도 모선을 비롯하여 외부대면부를 알아야 하며 새치기가 어떻게 조종되는가를 알아야 한다. 이러한 리유로 하여 처리장치의 기능을 규정하는데 필요한 몇가지 항목은 다음과 같다.

· 조작코드
· 주소지정방식
· 등록기
· I/O 모듈대면부
· 기억기모듈대면부
· 새치기처리구조

이 목록은 어느 정도 완성된것이다. 1~3 까지의 항목은 명령모임에 의하여 결정된다. 항목 4 와 5 는 전형적으로 체계모선에 의하여 결정된다. 항목 6 은 일부는 체계모선에 의하여 정의되며 일부는 처리장치가 조작체계에 제공하는 지원의 형태로 정의된다.

이 6 개의 항목의 목록을 처리장치에 대한 기능요구라고 말한다. 이것들은 처리장치가 무엇을 해야 하는가를 결정한다. 이에 대해서는 제 2 편과 제 3 편에서 론의하였다. 이제 이 기능이 어떻게 수행되며 처리장치안의 여러개 요소들이 이 기능을 수행하기 위하여 어떻게 조종되는가 하는 문제에 대하여 고찰하려고 한다. 결국 처리장치의 조작을 조종하는 조종장치를 론의한다.

제 1 절. 마이크로조작

이미 프로그람의 집행에서 콤퓨터의 조작이 한주기당 하나의 기계명령으로 되여 있는 명령주기들의 련속으로 구성되여 있다는것을 보았다. 여기서 잊지 말아야 할것은 명령주기의 실행과정이 프로그람에서 규정된 명령의 작성순서와 꼭같지 않다는것이다. 그것은 분기명령이 있기때문이다. 여기서 론의하려는것은 명령의 실행**시간순서**이다.

매개 명령주기는 또 더 작은 단위들로 이루어 졌다. 이 세분화된 단위들은 꺼내기, 간접, 실행 그리고 새치기주기인데 이 가운데서 꺼내기와 실행은 늘 발생한다.

보통 조종장치를 설계하자면 이것들을 보다 더 세분화하여야 한다. 제 11 장에서 관흐름을 론의하면서 개개의 작은 주기는 몇개의 단계로 되여 있는데 이 단계의 개개는 또 처리장치등록기들로 되여 있다. 이 단계를 **마이크로조작**이라고 한다. 앞붙이 마이크로라는 말은 매개 단계가 매우 단순하며 대단히 작다는것을 의미한다. 그림 14-1 은 지금까지

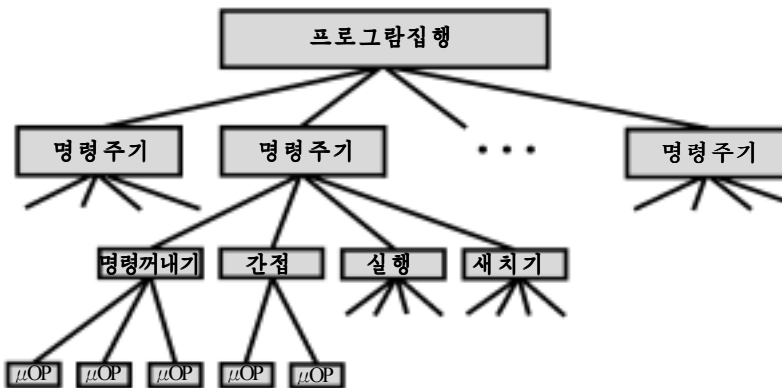



그림 14-1. 프로그람집행의 구성요소

론의한 여러가지 개념들의 관계를 보여 주었다. 종합적으로 말하면 프로그람의 실행은 명령의 순차적실행으로 구성되여 있다. 매개 명령은 보다 더 짧은 부분주기(레하면 꺼내

기, 간접, 실행, 새치기)로 이루어 진 명령주기기간에 실행된다. 매개 부분주기의 실행은 하나 혹은 그보다 더 짧은 마이크로조작으로 이루어 진다.

마이크로조작은 처리장치의 기능적이며 원리적인 조작이다. 이 절에서는 마이크로조작을 통하여 임의의 명령주기사건들이 어떻게 이러한 마이크로조작의 순서로 서술될수 있는가를 리해하기 위한 마이크로조작을 시험하려고 한다. 그 실례는 간단한것을 리용하였다. 이 장의 마지막에 마이크로조작의 개념이 조종장치설계에 어떻게 리용되는가를 보여 준다.

1. 꺼내기주기

먼저 매개 명령의 시작으로 되면서 명령을 기억기에서 꺼내는 꺼내기주기에 대하여 보기로 한다. 설명을 위하여 그림 11-7에서 보여 준 구성을 고찰해 보자. 등록기는 4개로 되여 있다.

- **기억주소등록기(MAR):** 체계모선의 주소선과 련결되여 있다. 이것은 읽기-쓰기조작을 위한 기억주소를 넣어 둔다.
- **기억완충등록기(MBR):** 체계모선의 자료선과 련결되여 있다. 여기에는 기억기에 기억시키거나 기억기로부터 읽기 위한 값들이 넣어 진다.
- **프로그람계수기(PC):** 꺼내려는 다음명령의 주소를 취한다.
- **명령등록기(IR):** 꺼낸 명령을 기억해 둔다.

처리장치등록기의 견지에서 꺼내기주기에 대한 동작과정을 보기로 하자. 그 실례를 그림 14-2에 주었다. 꺼내기주기의 시작에서 실행하려는 다음명령의 주소가 프로그람계수기(PC)에 있다. 이 경우에 주소는 11000100이다. 첫 단계는 PC에 있는 주소를 기억주소등록기(MAR)에 넣는다. 왜냐하면 MAR를 통해서만 체계모선의 주소선에 련결되기때문이다. 두번째 단계는 MAR에서 명령을 꺼내 오는 단계이다. 요구되는 주소는 주소모선으로 출구되며 이때 조종장치는 READ신호를 조종모선에 내보낸다. 그 결과가 자료모선우에 나타나며 기억완충등록기(MBR)에 넣어 진다. 이와 함께 다음명령을 꺼내기 위한

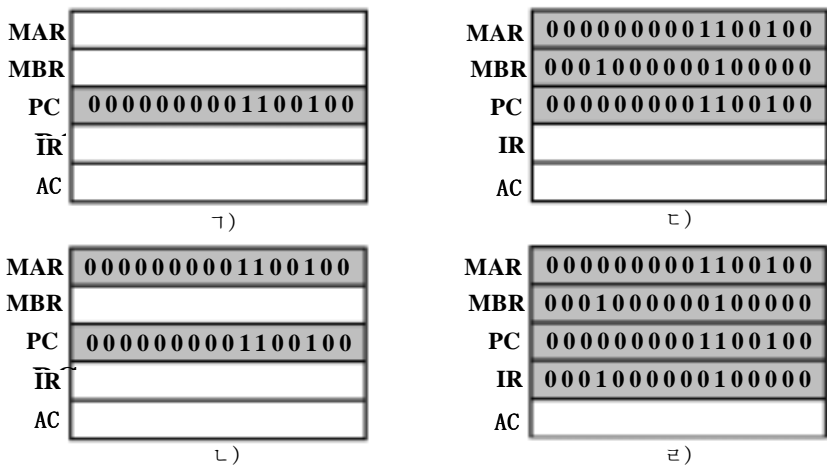



그림 14-2. 꺼내기주기동작과정

ㄱ- 1단계, ㄴ- 2단계, ㄷ-3단계, ㄹ- 4단계

준비로서 PC를 하나 증가시킨다. 그것은 이 두개의 동작(기억기로부터 단어읽기와 PC의 하나증가)이 서로 결합되여 있지 않기때문에 이 두 동작을 동시에 함으로써 시간을 절약할수 있다. 세번째 단계는 MBR의 내용을 IR에 옮기는 단계이다. 이것은 가능한 간접주기기간 MBR를 사용하기 위해서이다.

따라서 단순한 꺼내기주기는 실제적으로 3개의 단계와 4개의 마이크로조작으로 된다. 매개 마이크로조작은 자료를 기억기에서 읽어 내거나 기억기에 써넣는 자료옮김을 포함하고 있다. 이 옮김이 다른것과 관계되지 않는한 시간을 절약하기 위하여 이 여러단계들을 한단계에 수행할수 있다. 이 동작과정을 기호적으로 다음과 같이 쓸수 있다.

t_1:	MAR	←	(PC)
t_2:	MBR	←	기억기
	PC	←	(PC) + I
t_3:	IR	←	(MBR)

여기서 I는 명령길이와 관계된다. 이 과정에 대하여 여러가지로 설명을 붙일수 있다. 박자신호는 동기를 취하기 위한것이며 규칙적인 공간박자임풀스를 내보낸다. 매개 박자임풀스는 시간단위를 정의한다. 따라서 모든 시간단위들은 동일한 시간길이를 가진다. 매개 마이크로조작은 하나의 시간단위내에 수행된다. 기호 t는 시간단위를 의미한다. 말로 표현하면 아래와 같다.

· **첫번째 시간단위:** PC의 내용을 MAR에 옮긴다.
· **두번째 시간단위:** MAR에 의해 규정된 기억주소의 내용을 MBR에 옮긴다. 그리고 PC의 내용을 I만큼 증가시킨다.
· **세번째 시간단위:** MBR의 내용을 IR에 옮긴다.

여기서 참고할것은 두번째와 세번째마이크로조작은 둘 다 두번째 시간단위내에 일어난다는것이다. 세번째 마이크로조작은 꺼내기조작에 영향을 주지 않고 네번째 마이크로조작과 묶어 질수 있다.

t_1:	MAR	←	(PC)
t_2:	MBR	←	기억기
t_3:	PC	←	(PC) + I
	IR	←	(MBR)

마이크로조작의 그룹화는 다음과 같은 두가지 단순한 규칙에 따른다.

1. 일정한 동작순위를 따라야 한다. 즉 (MAR ← (PC))는 (MBR ← 기억기)보다 앞서야 한다. 그것은 기억기읽기조작이 MAR안에 있는 주소를 리용해야 하기때문이다.
2. 충돌을 피해야 한다. 같은 시간단위안에서는 같은 등록기에 대한 읽기쓰기를 하지 말아야 한다. 그것은 읽기쓰기를 하면 결과를 예측할수 없기때문이다. 례하면 마이크로조작들인 (MBR ← 기억기)와 (IR ← MBR)를 같은 시간단위안에 진행하지 않도록 해야 한다.

마지막으로 고려해야 할것은 마이크로조작중에 더하기연산을 포함한다는것이다. 순환성의 중복을 피하기 위하여 더하기는 ALU 가 수행한다. ALU 가 리용됨으로써 ALU의 기능과 처리장치의 구조에 따르는 마이크로조작이 보충적으로 포함되게 된다. 이 장

에서는 이에 대하여 론의한다.

여기에서 서술된 과정과 그림 3-5 의 다음에 있는 소제목들의 내용을 비교해 볼 필요가 있다. 이 그림에서는 마이크로조작이 무시되였으며 여기서는 명령주기의 보조주기를 수행하는데 마이크로조작이 필요하다는것을 보여 주었다.

2. 간접주기

일단 명령을 꺼낸 다음의 단계는 원천연산수를 꺼내는것이다. 우의 실례를 리용하여 직접 및 간접주소지정된 하나의 주소명령형식을 보기로 하자. 명령이 간접주소를 지적하자면 실행주기에 앞서 간접주기를 실행하여야 한다. 자료흐름은 그림 11-8 에서 보여 준것과 어느 정도 다르며 다음의 마이크로조작이 포함되여 있다.

t_1:	MAR	←	(IR(주소))
t_2:	MBR	←	기억기
t_3:	IR(주소)	←	(MBR(주소))

명령의 주소마당은 MAR 에로 전송된다. 이것은 그다음연산수의 주소를 꺼내는데 리용된다. 마지막으로 IR 의 주소마당은 간접주소라기보다 현재 직접주소를 가지고 있는 MBR 에 의하여 갱신된다.

IR 는 현재까지도 간접주소지정에 쓰이지 않으며 실행주기에 쓸수 있도록 준비된다. 새치기주기를 고찰하기 위하여 잠간 이 주기를 건너 뛴다.

3. 새치기주기

새치기주기의 끝에서 임의의 가능한 새치기가 발생하지 않았는가에 대한 검사를 진행한다. 새치기가 일어 나면 새치기주기를 실행한다. 이 주기의 성질은 한 기계로부터 다른 기계까지 크게 변한다. 그림 11-9 에서 보여 주는것처럼 간단한 동작과정을 아래에 제시하였다.

t_1:	MBR	←	(PC)
t_2:	MAK	←	보관주소
	PC	←	루틴주소
t_3:	기억기	←	(MBR)

첫 단계에서는 PC 의 내용이 MBR 에로 전송된다. 목적은 새치기로부터 복귀하는 경우를 위해 그 주소를 보관하자는데 있다. 그러면 MAR 에는 PC 의 내용을 보관할수 있게 되며 주소가 넣어 진다. 그리고 PC 에는 새치기처리루틴의 시작주소가 넣어 진다. 이 두 동작은 각각 간단한 마이크로조작으로 할수 있다. 대부분의 처리장치들은 여러가지 류형의 새치기준위를 제공하기때문에 보관주소와 루틴주소를 MAR 와 PC 에 전송되기전에 보관하기 위하여 하나 혹은 그이상의 보충적인 마이크로조작을 더 하여야 한다. 임의의 경우에 일단 이것이 진행되면 마지막단계로서 PC 의 이전값이 들어 있는 MBR 의 내용을 기억기에 기억시킨다. 처리장치는 다음명령주기를 시작할 준비를 한다.

4. 실행주기

꺼내기, 간접 그리고 새치기주기는 단순하면서도 예측할수 있다. 이들 개개는 작으면서 고정된 마이크로조작과정으로 되여 있으며 매 경우에 같은 마이크로조작이 매번 반복된다.

이것은 실행주기의 전 면모가 아니다. N 개의 서로 다른 조작코드를 가진 기계에는 N 개의 서로 다른 마이크로조작이 있게 된다. 몇개의 가상적인 실례를 고찰해 보기로 하자.

먼저 더하기 명령을 보기로 하자.

ADD R1, X

는 주소 X 의 내용을 등록기 R1 에 더한다. 그리고 다음의 마이크로조작과정이 있게 된다.

t_1: MAR ← (IR(주소))
t_2: MBR ← 기억기
t_3: R ← (R1) + (MBR)

이제 더하기명령이 들어 있는 IR 로부터 시작한다. 첫 단계에서 IR 안의 주소가 MAR 에 넣어 진다. 그다음 참조된 기억주소를 읽는다. 마지막으로 R1 와 MBR 의 내용이 ALU 에서 더해 진다. 역시 이것도 단순한 하나의 실례이다. 일부 보충된 마이크로조작들은 IR 에서 등록기참조를 요구하거나 일부 종단등록기에 ALU 의 입구 혹은 출구를 기억시키는 것을 요구할수 있다.

좀더 복잡한 실례를 두가지 더 들어 보자. 공통명령은 령이면 증가 및 이행명령이다.

ISZ X

주소 X 의 내용은 하나 증가된다. 결과가 령이면 다음명령을 건너 뛴다. 마이크로조작순서는 다음과 같다.

t_1: MAR ←(IR)(주소))
t_2: MBR ←기억기
t_3: MBR ←(MBR)+1
t_4: 기억기← (MBR)
IF ((MBK) = 0) then (PC ←(PC)+I)

여기에 도입된 새로운 특징은 조건작용이다. PC 는 (MBR)=0 이면 하나 증가된다. 이 검사와 작용은 하나의 마이크로조작처럼 실현할수 있다. 이 마이크로조작은 같은 시간동안에 수행되며 이 기간에 MBR 안에 있는 갱신된 값은 도로 기억기에 기억된다.

마지막으로 보조프로그람호출명령을 보기로 하자. 실례로 분기-보관 –주소명령을 보기로 하자.

BSA X

BSA 명령뒤에 있는 명령의 주소는 주소 X 에 써넣어 지며 주소 X+I 에서 실행을 계속한다. 기억된 주소는 복귀할 때 리용된다. 이것은 보조프로그람호출을 위한 간단한 수법이다. 다음의 마이크로조작은 이것을 충분히 보여 주고 있다.

t_1: MAR ←(IR(주소))

MBR ←(PC)

t_2: PC ←(IR(주소))

기억기←(MBK)

t_3: PC ←(PC) + 1

명령시작에서 pc 안의 주소는 다음명령의 주소이다. 이것은 IR 가 지적하는 주소에 기억된다. 다음주소는 또한 다음명령주기를 위한 주소를 제공하기 위하여 하나만큼 증가된다.

5. 명령주기

지금까지의 고찰을 통하여 명령의 매개 상을 요소적인 마이크로조작으로 분해할수 있다는것을 알수 있다. 실례에는 꺼내기, 간접 그리고 새치기주기에 대한 조작과정과 실행주기에 대한 조작과정의 매개 조작코드에 대한 마이크로조작과정을 주었다.

그림을 완성하기 위하여 마이크로조작과정을 함께 종합해 보는것이 필요하다. 이것을 그림 14-3 에서 진행하였다. 여기에 명령주기코드(ICC)라고 불리우는 2bit 등록기를 새로 가정하였다. ICC 는 주기의 견지에서 처리장치의 상태를 지적한다.

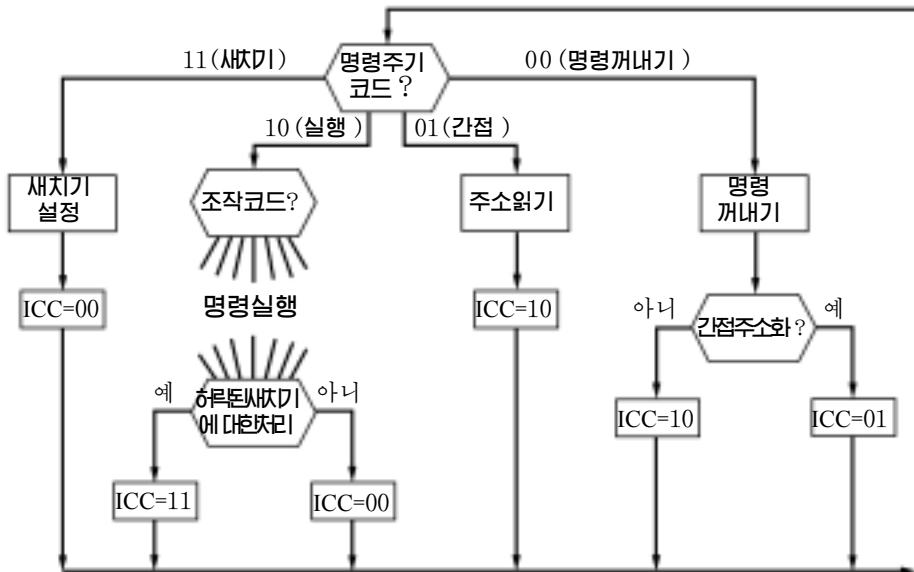



그림 14-3. 명령주기의 흐름도

`

00: 꺼내기

01: 간접

10: 실행

11: 새치기

네 주기의 개개의 끝에서 ICC 는 적당하게 설정된다. 간접주기는 늘 실행주기의 뒤를 따른다. 새치기주기는 항상 꺼내기주기뒤에 놓인다 (그림 11-5 참고). 실행과 꺼내기주기에 대하여 다음주기는 체계의 상태에 의존한다.

따라서 그림 14-3 의 흐름도는 명령실행과정과 새치기패턴에만 의존하는 마이크로조작의 완료과정을 보여 주고 있다. 물론 이것은 설명을 위하여 실례를 간단히 한것이다.

실제의 처리장치에 대한 흐름도는 보다 더 복잡하다. 일부 경우에 처리장치의 조작이 마이크로조작과정을 수행하는것으로 정리할수 있다고 말할수 있다. 조종장치가 이 과정을 어떻게 발생시키는가를 보기로 하자.

제 2 절. 처리장치의 조종

1. 기능에 대한 요구

앞절에서의 해석결과에 따라서 처리장치의 기동과 기능을 마이크로조작이라고 하는 요소조작으로 분해하였다. 처리장치의 조작을 여러개의 기능준위로 줄임으로써 조종장치가 무엇을 해야 하는가 하는것을 정확히 정의할수 있다. 따라서 조종장치가 수행해야 할 기능을 조종장치에 대한 기능요구라고 한다. 이 기능요구에 대한 정리는 조종장치의 설계와 실현을 위한 기초이다.

가까운 정보에 기초하여 다음의 3단계처리와 조종장치의 특징을 규정한다.

1. 처리장치의 기본요소들을 규정한다.
2. 처리장치가 수행할 마이크로조작을 서술한다.
3. 마이크로조작이 수행될수 있게 조종장치가 해야 할 기능을 규정한다.

1단계와 2단계는 이미 수행되였다. 그 결과를 종합해 보기로 하자. 우선 처리장치의 기본기능요소들은 다음과 같다.

- ALU
- 등록기
- 내부자료통로
- 외부자료통로
- 조종장치

이상의 고찰을 통하여 이것이 비교적 완성된 목록이라는것을 확신할수 있다. ALU는 콤퓨터의 기본요소이다. 등록기들은 처리장치에 내부자료를 기억시키는데 리용된다. 일부 등록기들은 명령실행과정을 관리하는데 필요한 상태정보(례하면 프로그람상태단어)를 보관한다. 다른 요소들은 ALU, 기억기, 입출력모듈에 입력하거나 출력할 자료를 기억해 둔다. 내부자료통로는 등록기들사이와 등록기와 ALU사이에 자료를 주고 받는데 리용한다. 외부자료통로는 흔히 체계모선을 통해서 등록기를 기억기와 입출력모듈과 련결시킨다. 조종장치는 처리장치내에서 조작이 생기도록 한다.

프로그람의 실행은 이 처리요소들에 대한 조작으로 이루어 진다. 우리가 이미 알고 있는것처럼 이 조작은 또한 마이크로조작과정으로 구성된다. 제 14장의 제 1절에서 보여준바와 같이 독자들은 모든 마이크로조작이 다음과 같은 분류가운데서 어느 한 종류에 속한다는것을 알고 있어야 한다.

- 자료를 어느 한 등록기로부터 다른 등록기에로 전송
- 자료를 등록기로부터 외부대면부(례하면 체계모선)에로 전송
- 자료를 체계모선으로부터 외부대면부에로 전송
- 입출구를 위한 등록기를 리용하여 산수 및 론리연산의 수행

명령모임안의 모든 명령을 실행하기 위한 모든 마이크로연산을 비롯하여 하나의 명령주

기안에 실행해야 할 마이크로연산들은 이와 같이 분류된다.

조종장치가 어떤 기능을 수행하는가에 대하여 좀 더 명백히 하는것이 필요하다. 조종장치는 두가지 기본과제를 수행한다.

- **순서작성:** 조종장치는 처리장치로 하여금 실행하려는 프로그람에 기초한 적합한 순서로 마이크로조작을 수행하게 한다.
- **실행:** 조종장치는 매개 마이크로조작이 수행되게 한다.

앞에서 설명한 내용들은 조종장치가 무엇을 해야 하는가 하는 기능에 대하여 해설한것이다. 조종장치를 어떻게 조작하는가 하는 근본열쇠는 조종신호에 있다.

2. 조종신호

지금까지는 처리장치를 구성하는 요소들과 수행해야 할 마이크로조작들에 대하여 정의하였다. 조종장치가 그 기능을 수행하기 위하여서는 체계의 상태를 결정하는 입구와 체계의 동작을 조종하기 위한 출구를 가지고 있어야 한다. 이것을 조종장치의 외적특성이라고 한다. 내부적으로 조종장치는 순서작성과 실행기능을 수행하기 위한 론리를 가지고 있어야 한다. 이 책에서는 제 14 장의 제 3 절과 제 15 장에서 조종장치의 내적조작을 주로 설명한다. 이 절의 뒤에서 조종장치와 처리장치의 다른 요소들사이의 호상관계를 고찰한다.

그림 14-4 는 모든 입구와 출구를 다 보여 준 조종장치의 일반모형이다. 그 입구는 다음과 같다.

- **박자:** 이것은 조종장치가 《시간을 맞추기》위한것이다. 조종장치는 매 박자임풀스마다 하나의 마이크로조작(혹은 동시적인 여러개의 마이크로조작)을 수행한다. 이것을 때로는 처리장치주기시간 또는 박자주기시간이라고 한다.
- **명령등록기:** 현행명령의 조작코드를 리용하여 실행주기기간 수행해야 할 마이크로조작코드를 결정한다.
- **기발:** 이것들은 처리장치의 상태와 선택한 ALU 조종의 결과를 결정하기 위하여 종장치에서 필요하다. 례하면 조건이 령이면 증가 및 이행하는 명령.

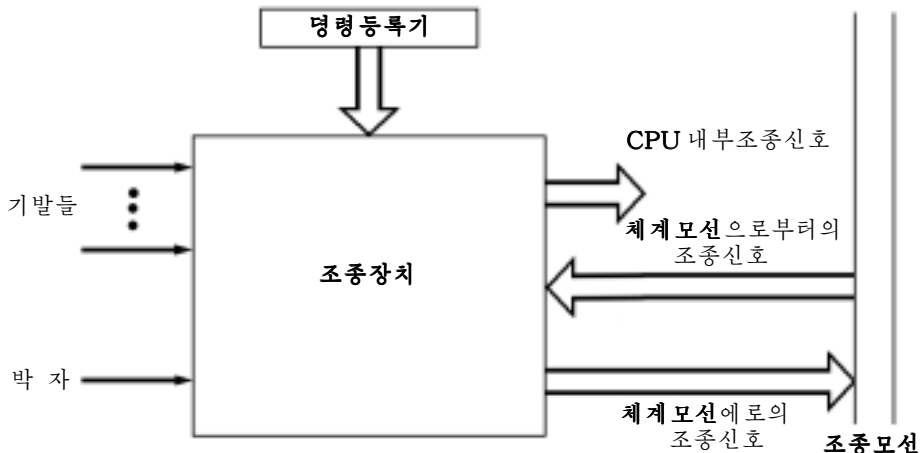



그림 14-4. 조종장치의 모델

- **조종모선으로부터 입구하는 조종신호:** 체계모선의 한 부분인 조종모선은 조종장치에 새치기신호와 응답신호와 같은 조종신호를 보낸다.

출구신호는 다음과 같다:

- **처리장치내부의 조종신호:** 여기에는 두가지 류형이 있는데 하나는 자료를 어느 한등록기로부터 다른 등록기에로 옮기게 하는 조종신호이며 다른 한가지는 규정된 ALU 의 기능을 능동으로 하기 위한것이다.
- **조종모선에로 출구하는 조종신호:** 이것도 역시 두가지 류형이 있는데 하나는 기억기를 조종하기 위한 조종신호이며 다른 하나는 입출구모듈을 조종하기 위한 조종신호이다.

이 그림에 새로 들어 간 부분은 조종신호이다. 조종신호에는 세가지 류형이 쓰이고 있는데 그것들로는 ALU 기능을 능동으로 만들기 위한 조종신호들과 자료통로를 능동으로 만들기 위한 조종신호, 외부체계모선 혹은 다른 외부대면부를 조종하기 위한 조종신호이다. 이 신호들은 모두 개별적인 론리문회로에 2진신호로서 직접 작용한다.

여기서 다시 꺼내기주기를 통하여 조종장치가 어떻게 조종을 유지하는가를 보기로 하자. 조종장치는 꺼내기주기가 명령주기안의 어디에 있는가를 추적한다. 주어 진점에서 꺼내기주기가 다음에 수행되게 된다는것을 찾아 낸다. 첫 단계는 PC 의 내용을 MAR 에로 전송한다. 조종장치는 이렇게 하기 위하여 PC 의 비트들과 MAR 의 비트들사이의 문회로를 여는 조종신호를 능동으로 만든다. 다음단계는 기억기로부터 MBR 에로 자료를 읽기 하며 PC 를 하나 증가시키는것이다.

- MAR 의 내용을 주소모선우에 태우기 위해 문회로들을 여는 조종신호
- 조종모선에 있는 기억읽기조종신호
- 자료모선의 내용을 MBR 에 기억시키기 위해 문회로들을 여는 조종신호
- PC 의 내용에 1 을 더하여 그 결과를 다시 PC 에 기억시키는 론리조작을 하는 조종신호

이에 따라서 조종장치는 MBR 와 IR 사이의 문회로를 여는 조종신호를 내보낸다.

이것으로의 조종장치는 간접주기든가 다음실행주기를 실행하겠는가 하지 않겠는가를 결정해야 하는것을 제외하고는 꺼내기주기를 완료한다. 이것을 결정하기 위하여 IR 를 검사하여 간접기억참조가 이루어 졌는가 아닌가를 식별한다.

간접주기와 새치기주기도 이와 류사하게 동작한다. 실행주기인 경우 조종장치는 조작코드에 대한 검사를 진행하며 그에 기초하여 실행주기기간 어느 마이크로조작과정을 수행하겠는가를 결정한다.

3. 조종신호의 실례

조종장치의 기능을 설명하기 위하여 간단한 실례를 보기로 하자. 그림 14-5 에 그 실례를 주었다. 이것은 축적등록기가 하나인 간단한 처리장치에 대한 실례이다.

부분들사이의 자료통로를 표시하고 있다. 조종장치로부터 출력되고 신호에 대한 조종통로는 표시하지 않았지만 조종신호의 끝은 C_i 라는 표식을 붙이고 동그라미로 표시하였다. 조종장치는 박자신호와 명령등록기값,기발을 입구로 받아 들인다. 매 박자주기마다 조종장치는 이 입구값들을 전부 받아 조종신호를 만들어 내보낸다. 조종신호는 세곳으로 출구된다.

- **자료통로:** 조종장치는 내부의 자료흐름을 조종한다. 례하면 명령을 꺼내는 경우 기억완충등록기의 내용은 명령등록기에로 전송된다. 매개 통로에 대한 조종은 문회로가 한다(그림에서 동그라미로 표시되여 있다). 조종장치로부터 출구되는 조종신호는 자료가 통과하도록 문회로를 연다.

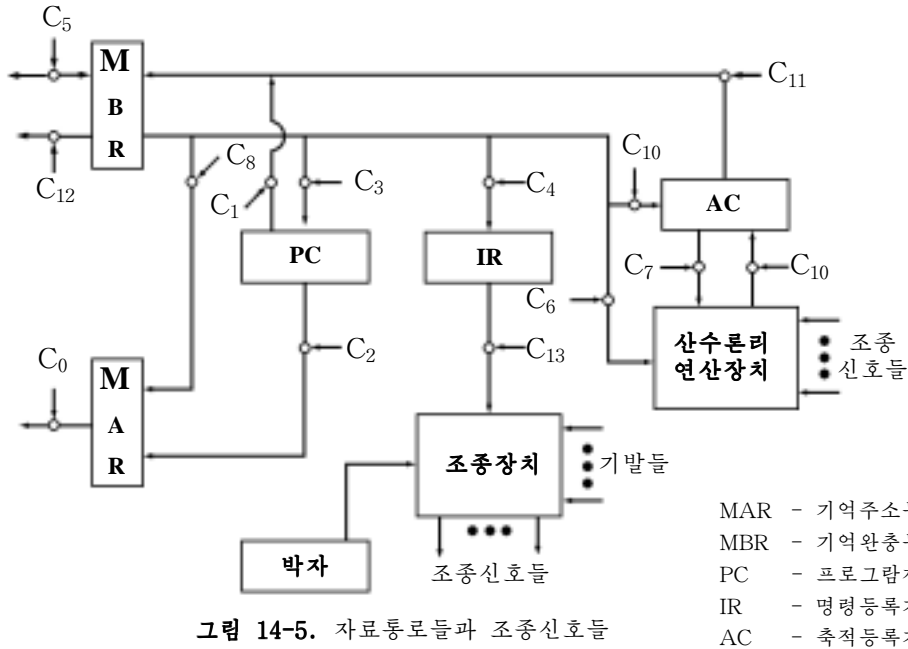



그림 14-5. 자료통로들과 조종신호들

- **ALU(산수-론리연산장치):** 조종장치는 조종신호들을 리용하여 ALU에 대한 조작을 진행한다. 이 신호들은 ALU 내의 여러개의 론리장치들과 문회로들을 능동으로 만든다.
- **체계모선:** 조종장치는 체계모선우의 조종신호선 례하면 기억 READ 신호에 조종신호를 내보낸다.

조종장치는 명령주기안에 있는 의미를 유지하고 있어야 한다. 조종장치는 이 의미를 유지하면서 모든 입구신호를 받아 들여 마이크로조작이 진행되도록 일련의 조종신호를 출력한다. 박자임풀스를 리용하여 사건의 동작과정을 동기화하며 신호준위의 안정화를 위하여 사건들사이의 동기화를 취한다. 표 14-1 은 앞에서 서술한 일부 마이크로조작과정을 위하여 필요한 조종신호를 보여 주고 있다. 간단히 하기 위하여 PC 의 내용을 증가시키며 PC 와 MAR 에 고정된 주소들을 넣기 위한 자료통로와 조종통로들은 표시하지 않았다.

이것으로 조종장치의 최소한도의 성질을 가늠할수 있다. 조종장치는 전체 콤퓨터를 동작시키는 기관이라고 말할수 있다. 조종장치는 실행된 명령의 뜻과 산수-론리연산결과의 상태(례하면 정의 값, 자리넘침)만을 안다. 조종장치는 처리된 자료나 얻어진 실제의 결과는 알지 못한다. 또한 조종장치는 처리장치에 입력되는 몇개의 조종신호와 체계모선에 출력되는 몇개의 조종신호들을 가지고 모든것을 조종한다.

표 14-1. 마이크로연산과 조종신호

마이크로연산	시　　간	능동조종신호
꺼내기	t_1: MAR ← (PC)	C_2
	t_2: MBR ← 기억기	C_5, C_R
	PC ← (PC) + 1	
	t_3: IR ← (MBR)	C_4
간접	t_1: MAR ←(IR(주소))	C_8
	t_2: MBR ← 기억기	C_5, C_R
	t_3: IR(주소) ← (MBR(주소))	C_4
새치기	t_1: MBR ← (PC)	C_1
	t_2: MAR ← 보관주소	
	PC ← 루틴주소	
	t_3: 기억기 ←(MBR)	C_{12}, C_W

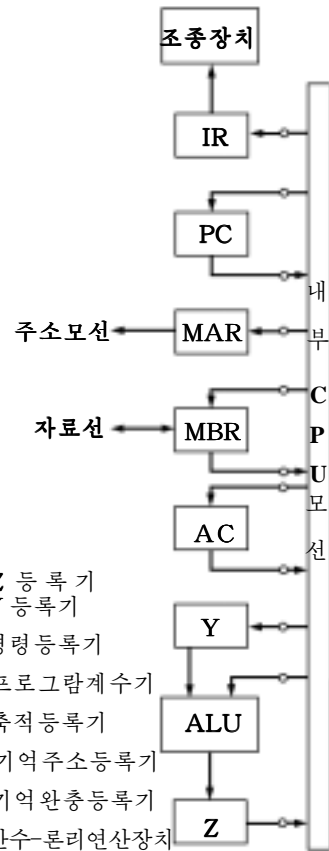
C_R - 체계모선에로의 조종신호읽기
C_W - 체계모선에로의 조종신호쓰기

4. 처리장치의 내부구성

그림 14-5에 여러개의 자료통로의 리용관계를 보여 주었다. 이러한 류형의 내부구성이 복잡하리라는것은 명백하다. 그림 11-2 에서 제기한것처럼 몇종류의 내부모선배치가 되여 있다.

처리장치의 내부모선을 리용하여 그림 14-5는 그림 14-6에서 보여 준것처럼 재배치할수 있다. 산수-론리연산장치와 처리장치의 등록기들은 하나의 내부모선에 다같이 련결되여 있다. 문회로와 조종신호들은 매개 등록기로부터 모선에로의 자료입출력을 조종한다. 보충된 조종신호들은 체계(외부)모선으로 나가거나 들어 오는 자료의 전송과 산수-론리연산장치에 대한 조작을 조종한다.

그림 14-6. 내부모선을 가진 CPU

Y 와 Z 라는 표식이 붙은 두개의 새 등록기가 내부구성에 더 포함되여 있다. 이것들은 산수-론리연산장치의 해당한 조작을 위하여 필요하다. 두개의 연산수가 포함된 연산을 수행할 때 하나는 내부모선으로부터 받지만 다른 하나는 다른 원천으로부터 받게 된다. AC 는 목적주소를 위하여 리용되지만 이것이 체계의 유연성에 제한을 주며 여러개의 일반목적등록기를 가진 처리장치처럼 동작할수 없다. 등록기 Y 는 다른 입구를 위한 림시기억기이다. ALU 는 내부기억기가 없는 조합회로이다. 따라서 조종신호가 ALU 의 기능을 능동으로 할 때 ALU 에 가해 진 입력자료는 출력자료로 변환된다. 그런데 ALU 의 출구는 직접 모선에 련결될수 없는데 왜냐

하면 이 출구가 입구에로 반결합될수 있기때문이다. 등록기 Z는 림시출구기억기로서의 기능을 수행한다. 이러한 배렬을 가지면 기억기로부터 출력된 값을 AC에 더하는 조작은 다음과 같은 단계를 거친다.

t_1: MAR ← (IR(address))
t_2: MBR ← 기억기
t_3: Y ← (MBR)
t_4: Z ← (AC)+(Y)
t_5: AC ← (Z)

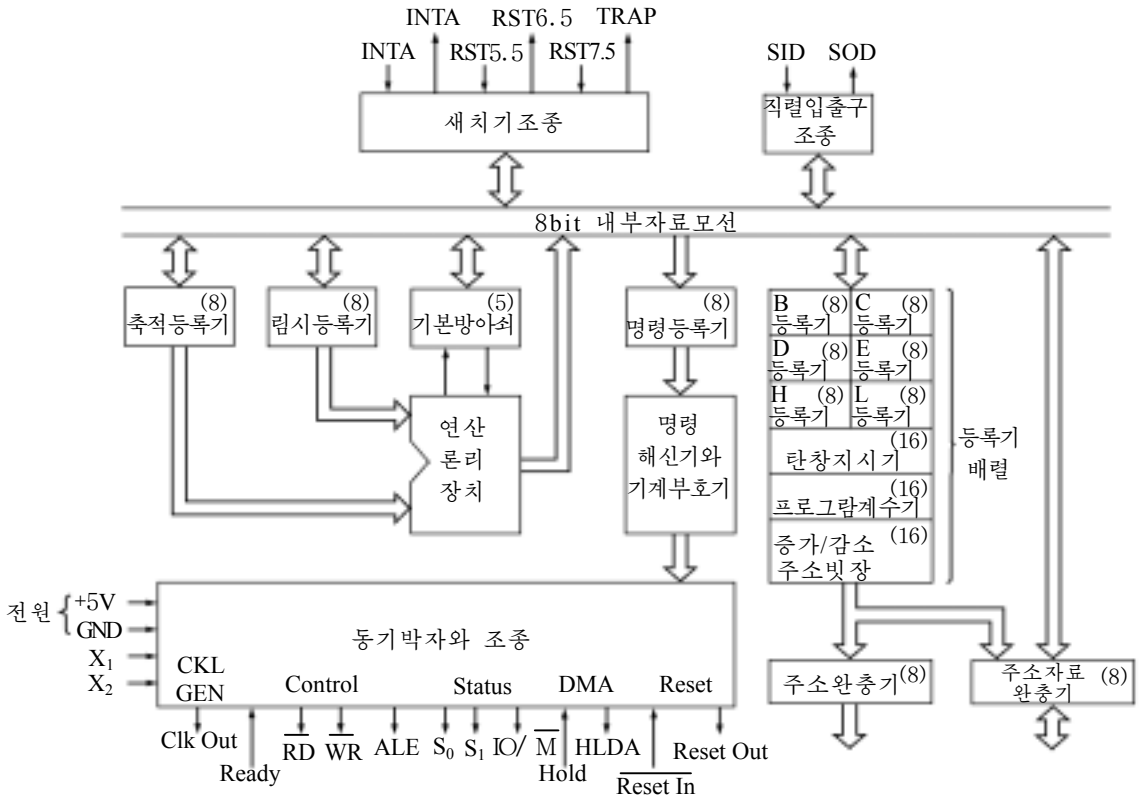
내부구성을 다르게 할수도 있지만 일반적으로는 몇가지 종류의 내부모선이나 내부모선묶음이 리용된다. 공통자료통로를 리용하면 내부련결설계와 처리장치의 조종을 간단히 할수 있다. 내부모선을 리용하는 다른 하나의 실천적의의는 공간을 절약하자는 것이다. 특히 ¼-inch 인 규소소편안에 넣어야 할 극소형처리장치에 대해서는 내부등록기들사이의 련결을 위한 공간을 최소화하여야 한다.

5. Intel 8085

이 장에서 소개된 일부 개념들을 설명하기 위하여 Intel 8085에 대하여 고찰해 보자. 그 내부구성을 그림 14-7에 보여 주었다. 몇가지 기본요소들은 다음과 같다.

그림 14-7. Intel 8085 CPU 블로크도

표 14-2. Intel 8085 외부신호

주소와 자료신호

높은 주소(A15-A8)

16bit 주소의 높은 자리 8

주소/자료(AD7-AD0)

16 주소의 낮은 자리 8 혹은 8 자료. 이 다중화는 단자들에 보관한다.

직렬입력자료(SID)

직렬로 전송하는 장치들에 알맞는 단일비트입력(한번에 한비트)

직렬출력자료(SOD)

직렬로 수신하는 장치들에 알맞는 단일비트출력

박자 및 조종신호

CLK(OUT)

체계박자. 매 주기는 하나의 T 상태를 표시한다. CLK 신호는 주변소편들에 공급되여 박자를 일치시킨다.

X1, X2

이 신호들은 외부수정편 혹은 내부박자발생기를 구동하는 다른 장치들에서 제공된다.

주소빗장허가(ALE)

기계주기의 첫 박자상태동안 발생하여 주변소편들이 주소선들에 기억하도록 한다. 이것은 주소모듈(례하면 기억기,I/O)이 주소화되고 있다는것을 인식하게 한다.

상태(S0, S1)

읽기 혹은 쓰기연산이 일어 나고 있는가를 가리키는데 리용되는 조종신호

IO/M

I/O 혹은 기억기모듈의 읽기 및 쓰기연산에 리용

읽기조종(RD)

선택된 기억기 혹은 I/O 모듈을 읽거나 자료모선이 자료전송에 리용되고 있다는것을 지시

쓰기조종(WD)

자료모선상의 자료가 선택된 기억기 혹은 I/O 위치에 써넣어 지고 있다는것을 지시

기억기 및 I/O 원시기호

Hold

CPU 가 외부체계모선의 조종과 리용을 포기할것을 요구. CPU 는 현재 IR 에 있는 명령실행을 끝내고 그다음 신호들이 CPU 에 의하여 조종, 주소 혹은 자료모선에 삽입되지 않는 동안 유지상태에 들어 간다. 유지상태동안 모선을 DMA 연산에 리용할수도 있다.

Hold Acknowledge(HOLDA)

이 조종장치출력신호는 HOLD 신호를 내보내여 현재 모선을 리용할수 있다는것을 지시

READY

보다 조속기억기 혹은 I/O 장치를 가진 CPU 를 동기화하는데 리용. 주소화된 장치가 READY 를 내보내면 CPU 는 입구(DBIN) 혹은 출구(WR)연산을 진행할수 있다. 이와 반대이면 CPU 는 장치가 준비될 때까지 대기상태에 들어 간다.

새치기관련신호

TRAP

새치기재시동(RST 7.5, 6.5, 5.5)

표 계속

새치기요구(INTR)

이 4개의 선들은 외부장치가 CPU를 중단시키는데 리용된다. CPU는 그것이 유지상태에 있거나 새치기가 불가능하게 된 경우에는 새치기요구신호를 접수하지 않는다. 새치기는 명령실행때에만 접수된다. 이 새치기들은 낮은 순위의 우선권을 가진다.

새치기응답

새치기에 긍정응답한다.

CPU 초기화

RESET IN

PC의 내용이 0으로 설정되게 한다. CPU는 위치 0에서 실행을 재개한다.

RESET OUT

CPU가 재설정되였음을 통보한다. 이 신호는 체계의 나머지부분을 재설정하는데 리용될수 있다.

전압과 접지

VCC

+5V 전원공급

VSS

접지

- **주소증가/감소빗장:** 탄창지시기 혹은 프로그람계수기의 내용을 하나 증가 또는 감소시킬수 있는 론리. 이것은 산수-론리연산장치를 리용하지 않으므로 시간을 절약한다.
- **새치기조종:** 이 모듈은 여러개 준위의 새치기신호를 조종한다.
- **직렬입출구조종:** 이 모듈은 한번에 1bit 씩 통신하는 장치와 결합된다.

그림 14-8. Intel 8085 단자배치

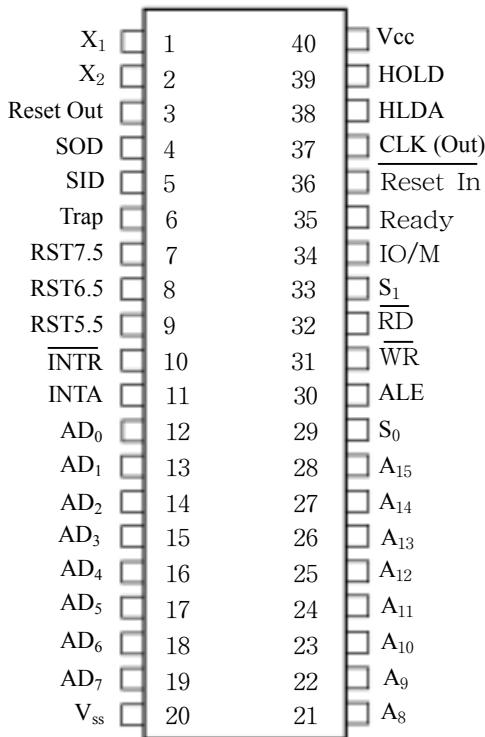



표 14-2는 8085에 입출력되는 외부신호를 서술한것이다. 이 신호들은 외부체계모선과 련결된다. 또한 8085 처리장치와 체계의 나머지구성요소들과의 결합을 한다(그림 14-8).

조종장치는 두가지 요소로 되여 있는데 하나는 명령해신기와 기계주기부호화모듈이고 다른 하나는 동기 및 조종모듈이다. 첫번째 모듈에 대한 론의는 다음절에서 한다. 조종장치에서 기본은 동기 및 조종모듈이다. 이 모듈은 박자신호를 포함하여 현행명령과 일부 외부조종신호들을 입구로 한다. 그의 출구는 처리장치내부의 다른 요소들에 가해 지는 조종신호와 외부체계모선에 출력되는 조종신호로 이루어 져 있다.

처리장치의 조작을 위한 동기는 박자신호에 의하여 진행되며 조종장치에 의하

여 조종된다. 매개 명령주기는 1~5 까지의 기계주기들로 이루어 지며 매개 기계주기는 또한 3~5 개의 상태주기로 되여 있다. 매개 상태주기는 한박자주기동안 지속된다. 이 기간 처리장치는 조종신호에 의하여 정해 진대로 하나 혹은 몇개의 마이크로연산을 동시에 수행한다.

기계주기의 수는 주어 진 명령에 대하여 고정되여 있다. 기계주기는 모선호출과 등가적으로 정의되여 있다. 따라서 명령에 대한 기계주기의 수는 처리장치가 외부장치와 통신하는 회수에 의존한다. 례를 들어 명령이 두개의 8bit 부분으로 구성되여 있다면 그때 두 기계주기는 명령을 꺼내는데 쓰인다. 그 명령이 1byte 의 기억기나 입출구조작인 경우 세번째 기계주기는 실행을 위한것이다.

그림 14-9 는 8085 의 동기실례를 주었는데 외부조종신호의 값을 보여 준다. 물론 같은 시간에 내부조종신호들은 내부자료전송을 조종하기 위하여 조종장치에서 발생된다. 시간선도는 OUT 명령에 대한 명령주기를 보여 주고 있다. 기계주기(M_1, M_2, M_3)에는 3 개의 주기가 요구된다. 첫 M1 기간에는 OUT 명령을 꺼낸다. 두번째 기계주기에는 명령의 절반을 꺼내는데 여기에는 출력하려는 입출력장치의 번호가 들어 있다. AC 의 내용은 세번째 주기기간에 자료모선을 통하여 선택된 장치에 써넣어 지게 된다.

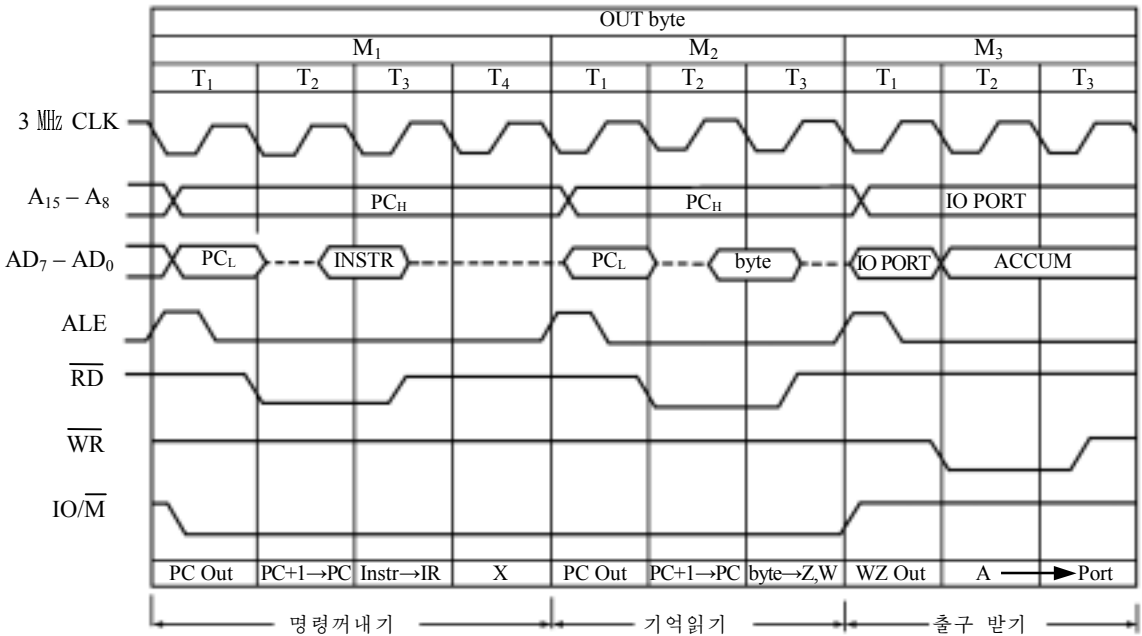



그림 14-9. 시간도형을 표시한 Intel 8085 OUT 처리장치

매 기계주기의 시작은 조종장치로부터 출력된 주소빗장가능신호(ALE)가 알려준다. 기계주기 M_1 의 동기상태 T_1 기간 조종장치는 이것이 기억조작이라는것을 알려 주는 IO/M 신호를 설정한다. 또한 조종장치는 PC 의 내용이 주소모선(A_{15}-A_8)과 주소-자료모선(AD_7-AD_0)에 출력되게 한다. 임풀스의 내림면에서 모선에 련결된 다른 모듈들은 주소를 기억한다.

동기상태 T_2기간 주소가 지적된 모듈은 지적된 기억주소의 내용을 주소-자료모선에 태운다. 이때 조종장치는 읽기상태를 의미하는 읽기조종신호(RD)를 설정한다. 그러나 T_3기간에는 자료를 주소모선으로부터 복사하기 위하여 대기한다. 이것은 모선우에 자료를 태우거나 신호준위를 안정시키기 위한 여유시간을 준다. 마지막상태인 T_4는 처리장치가 명령을 해신하는 기간으로서 모선무효상태(모선을 리용하지 않는 상태)이다. 나머지기계주기도 동작방식이 류사하다.

제 3 절. 장치적실현

지금까지 입구와 출구,기능의 견지에서 조종장치를 론의하였다.

여기서는 조종장치를 실현하는 문제에 대하여 고찰한다. 이를 위하여 여러가지 수법들이 리용되였는데 이들의 대부분은 아래의 두가지 실현방법가운데서 어느 한가지를 택하고 있다.

・장치배선론리실현
・마이크로프로그람실현

장치배선론리로 실현하는 조종장치는 본질에 있어서 조합회로이다. 조종신호는 그의 입구론리신호들은 출구론리신호로 변환한다. 이에 대한 기초를 이 절에서 취급하며 마이크로실현은 제 15 장에서 론의하기로 한다.

1. 조종장치입구

그림 14-4 는 지금까지 론의해 온 조종장치를 보여 주었다. 여기에서 주요입력은 명령등록기내용, 박자신호, 상태기발들과 조종모선신호들이다. 상태기발들과 조종모선신호들인 경우에 매개 개별비트들은 자기의 고유한 의미를 가진다(례하면 자리넘침). 그런데 다른 두개의 입구는 조종장치에 직접 리용할수 없다.

먼저 명령등록기에 대하여 고찰해 보자. 조종장치는 조작코드를 받아 서로 다른 명령에 대하여 서로 다른 동작(서로 다른 조종신호의 조합을 출력)을 수행한다. 조종장치의 론리를 간단히 하기 위하여서는 매개 조작코드에 대하여 단일론리입구가 있어야 한다. 이 기능은 해신기에 의하여 수행되며 해신기는 입구를 부호화하여 하나의 출력신호를 만든다. 일반적으로 해신기는 n 개의 2 진입구와 2^n개의 출구로 되여 있다. 2^n 개의 서로 다른 개개의 입력패턴은 오직 하나의 출구만을 능동으로 만든다. 표 14-3 이 그 실례이다. 조종장치를 위한 해신기는 가변길이조작코드들을 고려해야 하므로 이보다 더 복잡하게 된다. 해신기를 실현하는데 리용한 수자론리의 실례를 부록 I 에 주었다.

조종장치의 박자부분은 련속적인 임풀스를 반복적으로 내보낸다. 이것은 마이크로조작의 지속과정을 측정하는데 리용한다. 박자임풀스의 주기는 자료통로를 따라서 그리고 처리장치의 회로를 통하여 신호가 전파될수 있도록 충분히 길어야 한다. 그런데 조종장치는 하나의 명령주기내에 조종신호들을 서로 다른 시간단위로 출력한다. 따라서 T_1, T_2 등 기간에 리용되는 서로 다른 조종신호가 있으면 조종장

치에 가해 지는 입구로서는 계수기가 좋을것이다. 명령주기의 마지막에 조종장치는 T_1에서 다시 계수기를 초기화하기 위하여 여기에 반결합을 가한다.

이 두가지 구분에 기초하여 조종장치는 그림 14-10에서처럼 구성할수 있다.

2. 조종장치론리

조종장치에 대한 장치배선론리실현을 정의하기 위하여 아래에서 입력신호의 기능에 따라 출력조종신호를 만드는 조종장치의 내부론리에 대하여 고찰하자.

중요한것은 매개 조종신호에 대하여 진행해야 할것은 그 신호의 론리식을 입구기능에 따라 유도하는것이다. 이것은 실례를 들어 설명할수 있다. 그림 14-5에 보여 준 간단한 실례를 다시 고찰해 보자. 표 14-1에 명령주기의 4개의 상가운데서 3개 상에 필요한 마이크로조작과정과 조종신호들을 보여 주었다.

조종신호 C_5를 보기로 하자. 이 신호는 외부자료모선으로부터 MBR에로 자료를 읽기 위한것이다. 이것은 표 14-1에서 두번 리용된다는것을 알수 있다. 새로운 두개의 조종신호 P와 Q를 정의하자. 여기서 P와 Q는 다음과 같은 관계를 가진다.

PQ = 00　꺼내기주기
PQ = 01　간접주기
PQ = 10　실행주기
PQ = 11　새치기

그래서 C_5는 다음과 같은 론리식으로 정의된다.

$$C_5 = \overline{P} \cdot \overline{Q} \cdot T_2 + \overline{P} \cdot Q \cdot T_2$$

조종신호 C_5는 꺼내기와 간접의 두주기의 두번째 시간동안 유지된다.

표 14-3. 4개의 입구와 16개의 출구를 가지는 해신기

I1	I2	I3	I4	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15	O16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

표 계속

I1	I2	I3	I4	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15	O16
1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

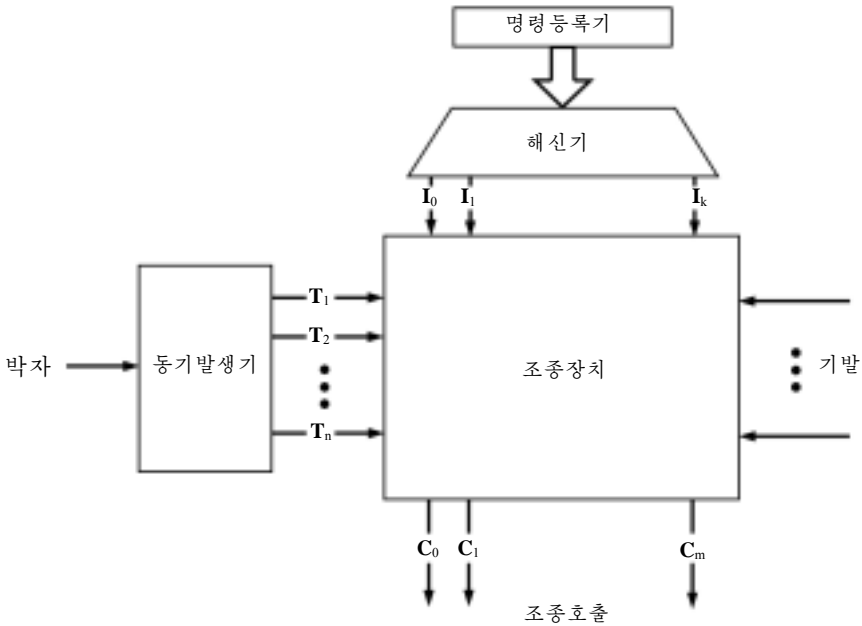



그림 14-10. 해신입구를 가진 조종장치

이 식은 완성된것이 아니다. C_5는 또한 실행주기기간 필요하다. 이 실례를 위하여 오직 기억기로부터 3개의 명령 LDA, ADD 그리고 AND만 읽기를 한다고 하자. 그러면 C_5는 다음과 같이 정의할수 있다.

$$C_5 = \overline{P} \cdot \overline{Q} \cdot T_2 + \overline{P} \cdot Q \cdot T_2 + P \cdot \overline{Q} \cdot (LDA + ADD + AND) \cdot T_2$$

이와 같은 처리는 처리장치에 의하여 발생된 조종신호에 따라 반복되여야 한다. 그 결과는 조종장치의 동작 즉 처리장치의 동작을 정의하는 론리식이다.

이것을 다 종합하여 조종장치는 명령주기의 상태를 조종한다. 이미 언급한것처럼 매개 부분주기(꺼내기, 간접, 실행, 새치기)의 끝에서 조종장치는 수행해야 할 다음부분주기를 규정하는 P와 Q의 해당한 값을 설정하여야 한다.

독자들은 현대의 복잡한 처리장치에서 조종장치를 정의하는데 필요되는 론리식의 전체를 만족하는 조합회로를 실현하는것은 매우 어려운 문제로 된다. 그 결과 **마이크로프로그람**이라고 하는 보다 더 간단한 방안이 흔히 리용되게 되였다. 이것은 다음장

에서 고찰한다.

참고문헌

[WARD90]과[MANO97]을 비롯한 일련의 참고문헌들에서 조종장치기능에 대한 기본원리를 취급하고 있다.

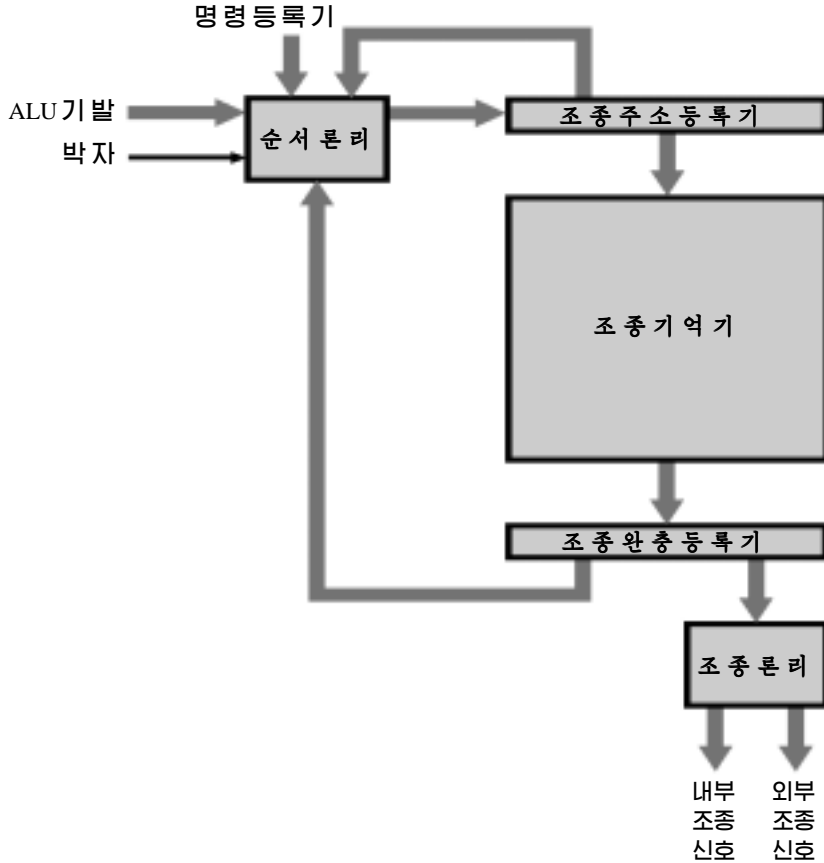
MANO97 Mano , M. Logic and Design Fundermen. Upper Saddle River, NJ: Prentice Hall, 1997

WARD90 Ward,S., and Halstead, R. Computation Structures. Cambridge,MA: MIT Press, 1990

련습문제

1. ALU 는 두개의 입력등록기를 더할수 있고 또 두 입력등록기의 비트들을 론리적으로 보수를 취할수 있는 가능이 있다고 하자. 그러나 덜기는 할수 없다. 수들은 2 의 보수형태로 기억된다. 조종장치가 덜기를 수행하기 위한 마이크로연산을 표로 작성하시오.
2. 다음의 명령들을 그림 14-5 에 있는 처리장치에 대한 표 14-1 과 같은 형식으로 마이크로조작과 조종신호들을 표시하시오 .
   - 축적등록기넣기,
   - 축적등록기쓰기,
   - 축적등록기더하기,
   - 축적등록기론리곱하기,
   - 이행,
   - AC = 0 이면 이행,
   - 축적등록기보수
3. 그림 14-6 의 모선과 산수-론리연산장치를 통한 전달지연이 각각 20,100ns 라고 하자. 등록기가 모선으로부터 자료를 복사하는데 필요한 시간은 10ns 이다. 다음의 경우에 요구되여야 할 시간은 얼마인가?
   ㄱ. 자료를 한 등록기로부터 다른 등록기로 전송하는데 요구되는 시간
   ㄴ. 프로그람계수기를 증가시키는데 요구되는 시간
4. 수가 다음과 같을 때 AC 에 수를 더하기 위하여 그림 14-6 의 모선구조가 요구하는 마이크로조작과정을 작성하시오.
   ㄱ. 즉시연산수,
   ㄴ. 직접주소연산수,
   ㄷ. 간접주소연산수
5. 탄창이 그림 9-14 에 보여 준것처럼 되여 있다. 다음의 경우에 대하여 마이크로조작과정을 작성하시오.
   ㄱ. 탄창에서 꺼내기
   ㄴ. 탄창에 넣기

제 15 장. 마이크로프로그람조종

◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

◆ 장치배선론리식조종장치실현의 또 한가지 방법은 조종장치의 론리가 마이크로프로그람에 의하여 규정되는 마이크로프로그람조종장치이다. 마이크로프로그람은 마이크로프로그람작성언어로 표시된 명령렬로 구성되여 있다.

◆ 마이크로프로그람조종장치는 마이크로명령에 의하여 동작순서를 결정할수 있으며 매개 마이크로명령을 실행하여 조종신호를 발생시킬수 있는 상대적으로 간단한 론리회로이다.

◆ 장치배선론리식조종장치에서와 같이 마이크로명령에 의하여 발생된 조종신호는 등록기전송과 ALU 조작을 하는데 쓰인다.

◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

마이크로프로그람이라는 말은 1950년대 초에 윌크스(M.V. Wilkes)에 의하여 처음으로 제안되였다[WILK51]. 윌크스는 조종장치를 구성하고 체계화하여 장치배선론리실현의 복잡성을 피할수 있는 설계방안을 제기하였다. 이 착상은 많은 전문가들의 흥미를 끌었으나 실현할수 없는 문제가 있었다. 그것은 고속이면서 상대적으로 값이 비싼 조종기억기가 필요했기때문이다.

마이크로프로그람기술의 이 문제는 1964년 2월에 나온 **자동자료처리**에 의하여 다시 고찰되였다. 그 시기에는 비마이크로프로그람체계가 광범히 리용되였으며 당시 론문들가운데서 하나의 론문인 [HILL64]에서는 마이크로프로그람의 미래에는《얼마간 구름이 끼여 있다. 생각하건데 마이크로프로그람작성자들 모두가 이에 대하여 연구하고 있다고 하더라도 그들중 누구도 그 기술에 흥미를 가지지 못하였다.》라는 전문가들의 견해를 종합하여 발표하였다.

이러한 사실은 바로 몇달후에 극적인 변화를 가져 왔다. IBM System/360이 그해 봄에 나왔는데 제일 큰 모듈은 마이크로프로그람화되였다. 비록 360계렬이 반도체 ROM의 기능을 훨씬 앞섰지만 마이크로프로그람의 우점을 고려하여 IBM은 이것을 받아 들이였다. 그중의 하나가 처리장치의 조종장치를 마이크로프로그람으로 실현한것이다.

제 1 절. 기본개념

1. 마이크로명령

조종장치는 합리적으로 간단한 장치처럼 보인다. 그러나 기본론리요소들을 호상 련결하여 조종장치를 실현하는것은 그리 쉬운 일이 아니다. 그 설계에는 마이크로조작순서를 위한 론리, 마이크로조작을 실행하기 위한 론리, 조작코드를 해석하기 위한 론리, ALU의 기발에 기초한 결심채택을 위한 론리가 포함되여 있어야 한다. 이러한 하드웨어부분을 설계하고 검사하는것은 어려운 문제이다. 더우기 설계는 상대적으로 변경시키기 어렵다. 례하면 새로운 기계명령을 더 보충하려고 하는 경우에 설계의 변경이 힘들다.

현대 CISC 처리장치에서 흔히 공통으로 리용되는 방안은 조종장치의 마이크로프로그람화를 실현하는것이다.

표 14-1을 다시 고찰해 보자. 조종신호를 리용하는 외에 매개 마이크로조작은 기호로 서술된다. 이 기호는 마치도 프로그람작성언어와 같다고 볼수 있다. 이것이 바로 **마이크로프로그람작성언어**이다. 매개 행은 한번에 수행할 마이크로조작들을 서술한것이며 일

명 **마이크로명령**이라고 한다. 이 명령렬을 **마이크로프로그람** 혹은 **험웨어**라고 한다. 험웨어라는 말은 마이크로프로그람이 하드웨어와 쏘프트웨어의 중간에 놓인다는데로부터 불여 진것이다. 하드웨어보다 험웨어를 설계하는것은 더 쉽지만 험웨어프로그람은 쏘프트웨어프로그람보다 작성하기가 더 힘들다.

마이크로프로그람의 개념을 조종장치를 실현하는데서 어떻게 리용할수 있는가? 매개 마이크로조작에 대한 개념과 조종장치에서 조종신호를 어떻게 발생시키는가를 고찰

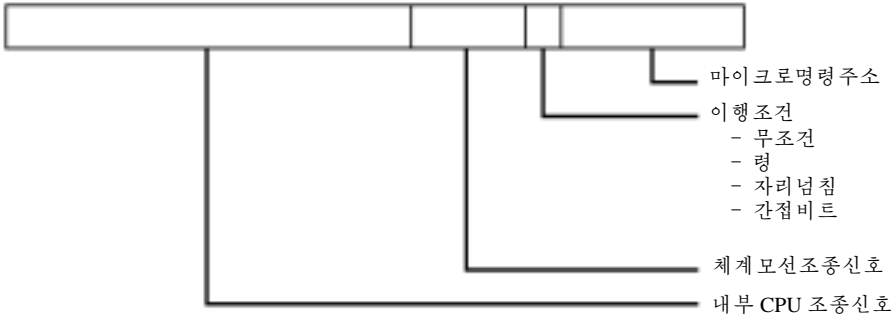
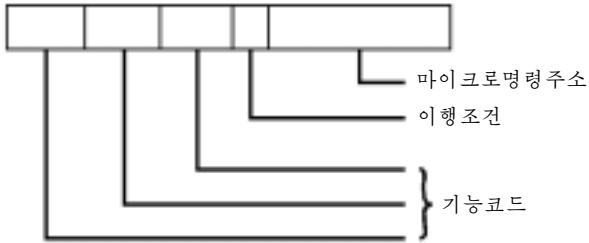



그림 15-1. 전형적인 마이크로명령형식
ㄱ-수평마이크로명령, ㄴ-수직마이크로명령

해 보자. 임의의 마이크로조작에 대하여 조종장치로부터 나오는 매개 조종신호선은 설정(ON)이나 차단(OFF)상태에 있다. 이 조건은 물론 매개 조종선을 2 진수로 표현할수 있게 한다. 그러므로 매 비트가 하나의 조종선을 의미하는 조종단어를 구성할수 있다. 따라서 매개 마이크로조작은 1과 0들의 서로 다른 패턴에 의하여 조종단어로 표현할수 있다.

조종장치에 의하여 수행되는 마이크로조작과정을 표현하는 조종단어의 렬을 가정하여 보자. 여기서 우리는 마이크로조작과정이 고정되여 있지 않다는것을 알아야 한다. 매

때로 간접주기가 있는 경우도 있고 없는 경우도 있으므로 조종단어를 기억기에 넣고 매개 단어에 유일주소를 할당한다. 그다음 매개 조종단어에 주소마당을 덧붙이고 이 주소가 일정한 조건이 참이면(례하면 기억기참조명령에서 간접이든가 1 이면) 집행될 다음조종단어의 위치를 지적하게 한다. 그외에 조건을 규정하기 위하여 몇개의 비트를 더 첨부한다.

이렇게 얻어 진 결과가 **수평마이크로명령**이며 그 실례를 그림 15-1 에서 보여 주었다. 마이크로명령 혹은 조종단어의 형식은 다음과 같다. 1bit 는 매개 내부처리장치조종선을 위한것이며 다음 1bit 는 매개 체계모선조종선을 위한것이다. 조건마당은 분기가 있거나 분기가 없는 경우에 다음에 집행할 마이크로명령의 주소를 가진 마당이 있다는것을 지적한다. 이러한 마이크로명령은 다음과 같이 해석된다.

1. 이 마이크로명령을 실행하기 위하여 1bit 로 지적된 조종선들은 모두 설정하며 0bit 로 지적된 조종선들은 모두 차단한다. 그 결과 조종신호들은 하나 혹은 여러개의 마이크로조작이 수행되게 한다.
2. 조건비트에 의하여 지적된 조건이 거짓이면 차례로 다음마이크로명령을 실행한다.
3. 조건비트에 의하여 지적된 조건이 참이면 실행하려는 다음마이크로명령이 주소마당에 의하여 지적된다.

그림 15-2 에 조종단어 혹은 마이크로명령들이 **조종기억기**안에 어떻게 배렬되는가를 보여 주었다. 매개 보조프로그람안의 마이크로명령들은 순서대로 실행된다. 매개 보조프로그람은 다음에 어디로 가야 하는가를 지적하는 분기 및 이행명령으로 끝난다. 여기에 특수한 실행주기보조루틴이 있는데 그 목적은 기계명령보조루틴(AND, ADD 등)가운데서 어느 하나만을 현재조작코드에 따라 다음에 집행하였는가 하는것을 표시하자는데 있다.

그림 15-2 의 조종기억기는 조종장치의 완료조작을 간단히 보여 준다. 이것은 매개 주기(꺼내기, 간접, 실행, 새치기)기간 수행되여야 할 마이크로조작과정을 정의하며 이 주기들의 순차를 정한다. 이 표기는 일부 콤퓨터에 쓰일 조종장치의 기능을 증명하기 위한 표기법이다. 실제는 이보다 더 많다. 다음으로 조종장치실현방법에 대하여 보기로 하자.

2. 마이크로프로그람조종장치

그림 15-2 의 조종기억기에는 조종장치의 동작을 서술한 프로그람이 들어 있다. 이 프로그람을 간단히 실행시키는것으로 조종장치를 실현할수 있다.

그림 15-2. 조종기억기의 구성

그림 15-3 은 이 실현을 위한 기본요소들을 보여 주었다. 마이크로명령들은 조종기억기안에 기억되여 있다. 조종주소등록기에는 읽어 들여야 할 다음 마이크로명령의 주소가 들어 있다. 마이크로명령은 조종기억기로부터 읽어 져 조종완충등록기에로 전송된다. 이 등록기의 왼쪽 부분(그림 15-1 ㄱ)은 조종장치로부터 나오는 조종신호선과 련결된다. 따라서 조종기억기로부터 마이크로명령의 읽기는 그 마이크로명령을 실행하는 것과 같다. 그림에서 보여 준 세번째 요소는 조종주소등록기를 읽어 읽기지령을 출구하는 순서장치이다.

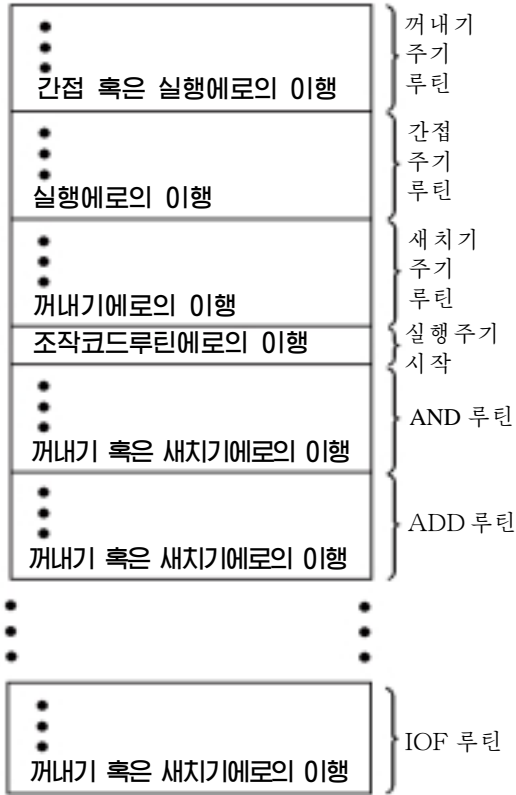



그림 15-4 에 보여 준것처럼 이 구조를 좀 더 구체적으로 살펴 보자. 그림 14-4 와 비교해 보면 조종장치는 여전히 같은 입구(IR, ALU, 상태기발, 박자)들과 출구(조종신호)들을 가지고 있다. 조종장치의 기능은 다음과 같다.

1. 명령을 실행하기 위하여 순서론리장치는 READ 지령을 조종기억기에 출력한다.
2. 그의 주소가 조종주소등록기에 규정되여 있는 단어를 조종완충등록기에서 읽는다.
3. 조종완충등록기의 내용은 순서론리장치에 대하여 조종신호와 다음주소정보를 발생시킨다.
4. 순서론리장치는 조종완충등록기와 ALU 기발들로부터 나오는 다음주소정보에 따라서 조종주소등록기에 새 주소를 넣는다.

이 모든것은 한박자동안에 진행된다.

마지막단계를 잘 처리하는것이 필요하다. 매개 마이크로명령의 끝에서 순서론리장치는 조종주소등록기에 새 주소를 넣는다. ALU 기발들과 조종완충등록기의 값에 따라 아래의 세가지 가운데서 어느 한가지 결론을 얻을수 있다.

• **다음명령꺼내기:** 조종주소등록기에 1을 더한다.
• **이행명령에 따라서 새로운 보조프로그람에로 이행:** 조종완충등록기의 주소마당을 조종주소등록기에 넣는다.

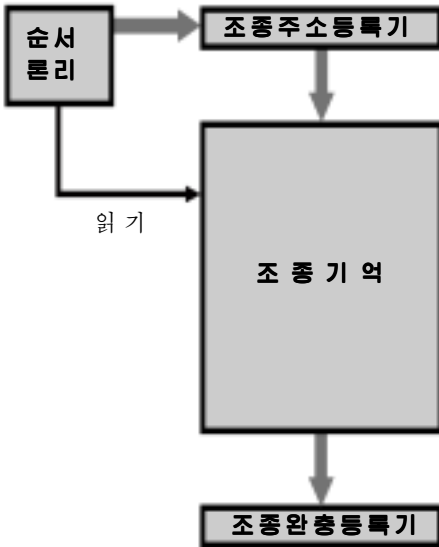



그림 15-3. 조종장치의 마이크로 구성방식

• **기계명령보조프로그람에로 이행:** IR 안의 조작코드에 따라서 조종완충등록기에 넣는다.

그림 15-4는 해신기라는 표식이 붙은 두개의 모듈을 보여 준다. 우에 있는 해신기는 IR의 조작코드를 조종기억주소로 변환한다. 아래의 해신기는 수평마이크로명령에서는 쓰지 않지만 **수직마이크로명령**에서는 쓰인다(그림 15-1 ㄴ). 우에서 언급된것처럼 수평마이크로명령에서 조종마당안의 매개 비트는 조종선에 련결된다. 수직마이크로명령에서 코드는 수행되여야 할 매개 동작을 규정하는데 리용되며(례하면 MAR ←(PC))해신기는 이 코드를 개별적인 조종신호로 변환한다. 수직마이크로명령의 우점은 수평마이크로명령보다 비트수가 훨씬 적다는것이다.

3. 윌크스조종

앞에서 언급된 바와 같이 윌크스는 1951년에 처음으로 마이크로프로그람조종의 리용을 제안하였다[WILK51]. 이 제안은 보다 더 구체적인 설계에로 발전하였다[WILK53]. 이 초기의 제안을 고찰해 보는것이 필요하다.

윌크스에 의하여 제안된 구성을 그림 15-5에 주었다. 이 체계에서 핵은 2극소자들로 구성된 행렬이다. 기계주기기간에 2 극소자행렬의 한개 렬이 임풀스에 의하여 능동으로 된다. 이것은 2극소자가 있는 곳에서 신호를 발생시킨다(그림에서 점으로 표시된 부분). 렬의 첫 부분은 처리장치의 동작을 조종하는 조종신호를 만든다. 두번째 부분은 다음 기계주기에서 동기될 렬의 주소를 발생시킨다. 따라서 행렬의 매개 렬은 하나의 마이크로명령에 대응하며 행렬의 배렬은 조종기억기기능을 수행한다.

박자의 시작에서 임풀스가 가해 진 렬의 주소는 등록기 I에 들어 간다. 이 주소는 다시 해신기에 입구되며 박자임풀스에 의하여 능동이 되면 대응하는 행렬에서 하나의 렬

을 능동으로 만든다. 조종신호에 따라서 명령등록기안의 조작코드라든가 임풀스가 가해진 렬의 두번째 부분이 주기기간 등록기 II를 통과한다. 그다음 등록기 II는 박자임풀스

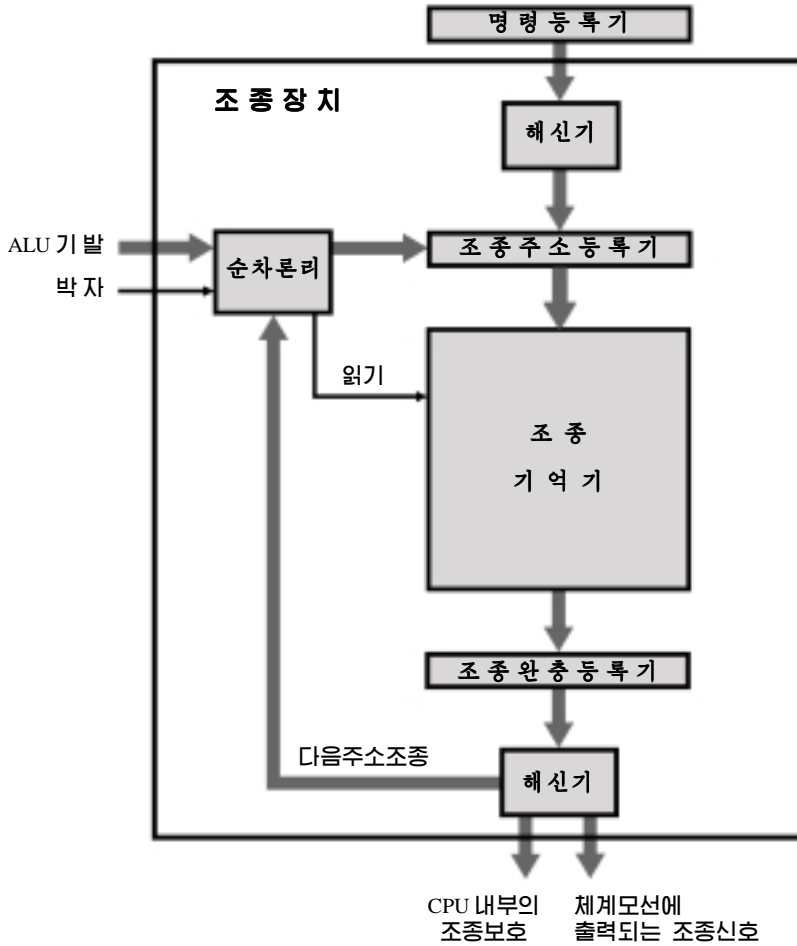



그림 15-4. 마이크로프로그람조종장치의 기능

에 의하여 등록기 I 에 입구된다. 박자임풀스가 계속 가해 짐에 따라 행렬의 렬이 능동으로 되며 등록기 II 로부터 등록기 I 에로 전송이 실현된다. 해신기가 조합회로이므로 두 등록기배렬이 요구되며 한 등록기만을 가지면 비안정조건이 생겨 출구가 박자주기기간 입구로 되게 된다. 이것은 그림 15-1 ㄱ에서 보여 준 수평마이크로프로그람방식과 매우 류사하다. 기본차이는 선행한 서술에서는 조종주소등록기가 다음주소를 만들기 위하여 하나만큼 증가된다는것이다. 윌크스방법에서는 다음주소가 마이크로명령안에 포함되여 있다. 분기하기 위하여 렬에는 그림에서 보여 준것처럼 조건신호(례하면 기발)에 의하여 조종되는 두개 주소가 있어야 한다.

이 방안을 제안한 후 윌크스는 기계의 조종장치를 실현하는데 이것을 리용하여 증명

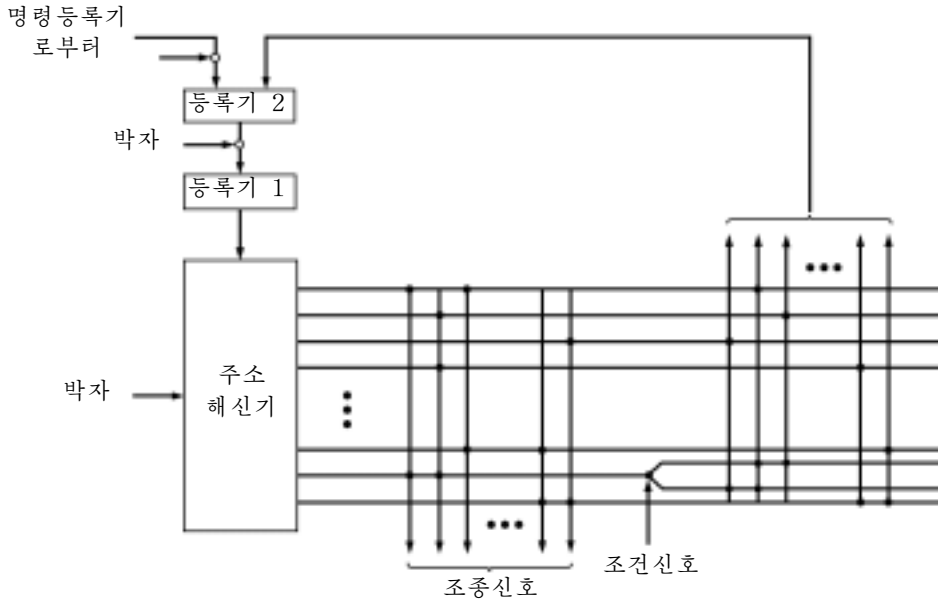



그림 15-5. 윌크스마이크로프로그람 조종장치

하였다. 마이크로프로그람처리장치의 설계로서 처음으로 알려 진 이 실례를 여기에서 다시한번 반복해 보는것이 마이크로프로그람의 기초원리를 인식하는데서 의의가 있을 것이다.

가상기계의 처리장치는 다음과 같은 등록기들을 가지고 있다.

A　축적등록기(곱해 질 수)
B　축적등록기(아래자리절반)
C　축적등록기(웃자리절반)
D　밀기등록기

또한 여기에는 3개의 등록기들과 조종장치에로의 호출만을 할수 있는 두개의 1bit기발이 있다. 그 등록기들은 다음과 같다.

E　기억기주소등록기(MAR)와 림시기억기로서 봉사
F　프로그람계수기
G　계수에 리용되는 다른 림시등록기

표 15-1은 이 실례에서 리용된 기계명령모임의 목록이다. 표 15-2는 조종장치를 실현하기 위하여 기호로 표현된 마이크로명령의 묶음일식이다. 체계를 완전히 정의하는데는 38개의 마이크로명령이 전부 있어야 한다. 첫번째 행은 매개 마이크로명령의 주소(렬번호)를 준다. 조작코드에 대응하는 이 주소들은 표식이 붙어 있다. 따라서

표 15-1. 윌크스의 기계명령모임실례

순서	기능
An	$Acc1 \leftarrow C(Acc) + C(n)$
Sn	$Acc1 \leftarrow C(Acc) - C(n)$
Hn	$Acc2 \leftarrow C(n)$
Vn	$Acc \leftarrow C(Acc2) \times C(n)$, 여기서 $C(n) \geqq 0$
Tn	$n \leftarrow C(Acc1), Acc \leftarrow 0$
Un	$n \leftarrow C(Acc1)$
Rn	$Acc \leftarrow C(Acc) \times 2^{-(n+1)}$
Ln	$Acc \leftarrow C(Acc) \times 2^{n+1}$
Gn	$C(Acc) < 0$ 이면 $n \leftarrow$ 흐름조종, $C(Acc) \geqq 0$ 이면 무시
In	입구기구에서 다음문자를 *n* 으로 읽기
On	출구기구 $\leftarrow C(n)$

주의: *Acc* = 축적기
Acc1 = 축적기의 웃자리절반
Acc2 = 축적기의 아래자리절반
n = 기억주소 *n*
C(X) = X 의 내용(X - 등록기 혹은 기억기주소)

더하기명령에 대한 조작코드(A)가 나타나면 5번째 위치에 있는 마이크로명령이 실행된다. 2 행과 3 행은 ALU 와 조종장치에 의하여 취해 지는 동작을 표시한다. 매개 기호표현은 조종신호(마이크로명령비트들)로 변환되여야 한다. 4행과 5행은 두개의 기발(방아쇠)들을 설정하고 리용하여야 한다. 4 행은 기발을 설정하는 신호를 규정한다. 례하면 C_S 는 기발번호 1 이 등록기안에 있는 수의 부호비트에 의하여 설정된다는것을 의미한다. 만일 5 행에 기발식별자가 있으면 그때 6 행과 7 행에는 리용해야 할 두개의 서로 다른 마이크로명령주소가 들어 있다. 다른 한편 6 행은 꺼내야 할 다음 마이크로명령의 주소를 규정한다.

명령 0～4 는 꺼내기주기를 이룬다. 마이크로명령 4 는 해신기에 조작코드를 넣어 꺼내려고 하는 기계명령에 대응하는 마이크로명령의 주소를 발생시킨다. 표 15-2 를 주의깊게 연구하면 조종장치의 완전한 기능을 알수 있다.

표 15-2. 윌크스의 마이크로명령실례

주의: *A, B, C,*……등은 산수연산 및 조종등록기장치의 각이한 등록기들을 의미한다. $D \leftarrow C$ 라는것은 스위치회로들이 *C* 등록기의 출구를 입구등록기 *D* 에 련결시켰다는것을 의미한다. $C \leftarrow (D + A)$ 라는것은 *A* 등록기의 출구가 가산장치(여기서 *D* 의 출구는 다른 입구들에 련결되였다.)의 한 입구에 련결되였으며 가산기의 출구가 *C* 등록기에 련결되였다는것을 의미한다. 인용문안의 수자기호 n 는 아래자리수자의 단위중에서 출구가 n 번째인 원천을 의미한다.

	산수연산부	조종등록기부	조건적방아쇠		다음마이크로명령	
			설정	리용	0	1
0		$G, E \leftarrow F$			1	

	산수연산부	조종등록기부	조건적 방아쇠		다음마이크로명령	
			설정	리용	0	1
1		$F\leftarrow$(‘1’$\leftarrow G$)			2	
2		$G\leftarrow$ 기억기			3	
3		$E\leftarrow G$			4	
4		해신기$\leftarrow E$			—	
A 5	$D\leftarrow C$				16	
S 6	$D\leftarrow C$				17	
H 7	$B\leftarrow$ 기억기				0	
V 8	$A\leftarrow$ 기억기				27	
T 9	기억기$\leftarrow C$				25	
U 10	기억기$\leftarrow C$				0	
R 11	$D\leftarrow B$	$G\leftarrow E$			19	
L 12	$D\leftarrow C$	$G\leftarrow E$			22	
G 13		$G\leftarrow E$	(1)C_5		18	
I 14	기억기$\leftarrow$ 입구				0	
O 15	출구 $\leftarrow$기억기				0	
16	$C\leftarrow$ (D +기억기)				0	
17	$C\leftarrow$ (D −기억기)				0	
18				1	0	1
19	$B(R)\leftarrow D$ *	$E\leftarrow$(‘1’$-G$)			20	
20	$D\leftarrow C$		(1)E_5		21	
21	$C(R)\leftarrow D$			1	11	0
22	$C(L)\leftarrow D$ †	$E\leftarrow$(‘1’$-G$)			23	
23	$D\leftarrow B$		(1)E_5		24	
24	$B\leftarrow C$			1	12	0
25	$B\leftarrow$ ‘0’				26	
26	$C\leftarrow B$				0	
27	$C\leftarrow$ ‘0’	$E\leftarrow$ ‘18’			28	
28	$D\leftarrow B$	$G\leftarrow E$	(1)B_1		29	
29	$B(R)\leftarrow D$	$E\leftarrow$(‘1’$-G$)			30	
30	$D(R)\leftarrow C$		(2)E_5	1	31	32
31	$C\leftarrow D$			2	28	33
32	$C\leftarrow(D+A)$			2	28	33
33	$D\leftarrow B$		(1)B_1		34	
34	$B(R)\leftarrow D$				35	
35	$D(R)\leftarrow C$			1	36	37
36	$C\leftarrow D$				0	
37	$C\leftarrow(D-A)$				0	

*오른쪽밀기. 산수연산장치에서 스위치회로들은 오른쪽밀기마이크로연산기간 등록기 C의 맨 아래자리수자가 등록기 B의 맨 웃자리위치에 놓이고 등록기 C의 맨 웃자리수자(부호수자)가 반복되도록(결국 부수인 경우 보정한다) 배렬한다.

†왼쪽밀기. 스위치회로들은 우와 류사하게 왼쪽밀기마이크로연산기간 B의 맨 웃자리수자를 C등록기의 맨 아래자리위치로 넘기도록 배렬한다.

4. 우점과 결함

조종장치의 실현에 마이크로프로그람을 리용하면 조종장치의 설계를 간단히 할수 있는 원리적인 우점이 생긴다. 즉 설계를 보다 값 눅게, 보다 오유가 적게 실현할수 있다.

하드웨어배선론리식조종장치는 많은 명령주기를 가지는 마이크로조작을 통하여 순서를 짜야 하므로 론리가 복잡해 진다. 반대로 마이크로프로그람조종장치의 해신기들과 순서론리장치는 론리가 대단히 간단하다.

마이크로프로그람장치의 기본결함은 하드웨어배선론리장치보다 속도가 뜬것이다. 그럼에도 불구하고 마이크로프로그람은 실현이 쉽기때문에 현대 CISC 조종장치실현에서 지배적인 기술로 되고 있다. 보다 더 단순한 형식을 가진 RISC처리장치들은 대표적으로 하드웨어배선론리조종장치를 리용하고 있다. 이것을 통하여 마이크로프로그람방식에 대한 보다 더 구체적인 리해를 가질수 있다.

제 2 절. 마이크로명령순서화

마이크로프로그람조종장치에 의하여 수행되여야 할 두가지 과제는 다음과 같다.

- **마이크로명령순서화:** 조종기억기로부터 다음마이크로명령을 꺼낸다.
- **마이크로명령실행:** 마이크로명령실행에 필요한 조종신호를 발생시킨다.

조종장치를 설계함에 있어서 이 두 과제는 함께 고려되여야 하는데 그것은 마이크로명령의 형식과 조종장치의 동기에 영향을 미치기때문이다. 이 절에서는 순서화를 위주로 설명하고 형식화와 동기에 대한 문제는 될수록 적게 론의한다.

1. 설계에서 고려할 점

마이크로명령순서화기술의 설계에서는 두가지 문제를 고려해야 하는데 하나는 명령의 크기이고 다른 하나는 주소발생시간이다. 첫번째 문제는 조종기억기의 크기를 최소화하여 그 부분가격을 줄이자는것이며 두번째 문제는 될수록 마이크로명령이 빨리 실행되게 하자는것이다.

마이크로프로그람실행에서 실행해야 할 다음마이크로명령의 주소는 아래의 분류가운데서 어느 하나에 의하여 얻어 진다.

- 명령등록기에 의하여 결정되는것
- 다음주소
- 분기

첫번째것은 명령꺼내기를 한 다음 명령주기당 한번만 생긴다. 두번째것은 대부분의 설계에서 가장 공통적인것이다. 그러나 순차호출인 경우 최적화할수 없다. 조건 및 무조

건명령에 따르는 분기는 마이크로프로그람에서 필수적인 부분이다. 더우기 마이크로명령 순서는 짧은것이 특징인데 3~4 개의 마이크로명령마다 하나의 분기가 있다[SIEW82]. 따라서 집약화되고 시간효률이 높은 마이크로명령분기를 위한 기술이 중요하다.

2. 순서화기술

현재의 마이크로명령, 조건기발과 명령등록기의 내용에 기초하여 다음마이크로명령에 대하여 조종기억주소를 발생시켜야 한다. 이를 위하여 여러가지 기술이 리용되였다. 이 기술들은 그림 15-6 부터 그림 15-8 에서 제시한것처럼 3 가지 일반적인 부류로 나눌수 있다. 이 마이크로명령안의 주소정보의 형식에 기초하여 이것을 분류하였다.

- 두개 주소마당
- 하나의 주소마당
- 가변형식화

제일 간단한 방법은 매개 마이크로명령에 두개 주소마당을 할당하는것이다. 그림 15-6 에 이 정보가 어떻게 리용되는가를 보여 주었다. 다중선택기는 주소마당과 명령등록기내용을 더하여 목적주소를 얻기 위하여 리용된다. 주소선택입구에 따라서 다중선택기는 조작코드라든가 두개의 주소가운데서 하나를 조종주소등록기(CAR)에 전송한다. 계속하여 CAR 는 해신되여 다음의 마이크로명령주소를 만든다. 주소선택신호는 분기론리모듈에 의하여 제공되는데 분기론리모듈의 입구는 조종장치기발들과 마이크로명령의 조종부에서 나오는 비트들로 되여 있다.

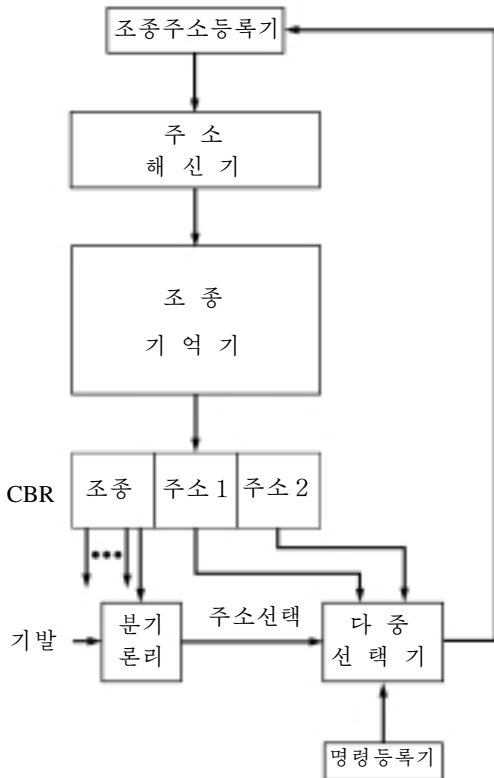



그림 15-6. 분기조종론리, 2 개의 주소구역

두 주소방식은 비록 간단하지만 다른 방식에 비하여 마이크로명령의 길이가 긴것이다. 일부 론리를 보충하여 이것을 절약할수 있다. 흔히 쓰이는 방식은 하나의 주소마당방식이다(그림 15-7). 이 방식을 리용하여 다음주소에 대한 선

택을 다음과 같이 한다.

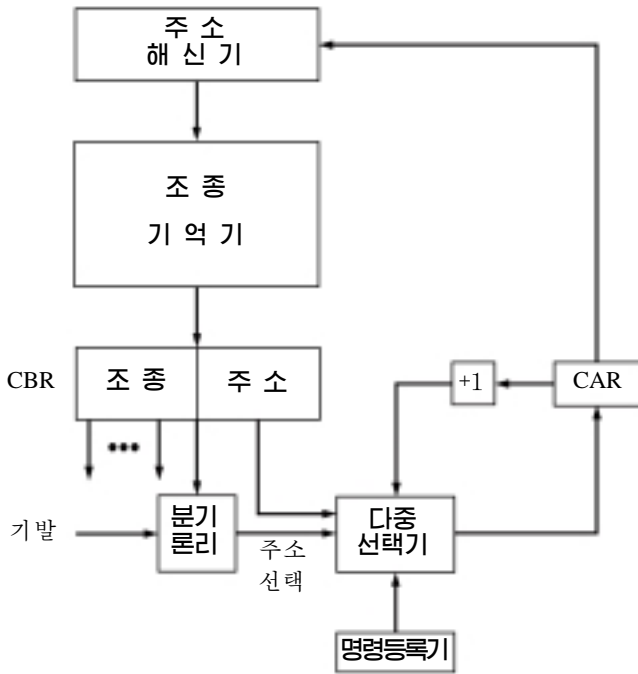



그림 15-7. 분기조종론리, 한개의 주소구역

- 주소마당
- 명령등록기해신
- 다음순차주소

주소선택신호들은 어느것이 선택되였는가를 결정한다. 이 방식은 주소마당의 수를 하나로 줄인다. 그런데 참고해야 할것은 일반적으로 주소마당은 리용되지 않는다는것이다. 따라서 마이크로명령에 대한 코드화에서 일부 비효과적인것이 있다.

또 다른 방식은 완전히 서로 다른 두개의 마이크로명령형식(그림 5-8)을 제기하고 있다. 한비트는 형식화가 리용되고 있다는것을 지적한다. 어떤 형식화에서 나머지비트들은 조종신호를 능동으로 하는데 쓰인다. 다른 형식화에서 일부 비트들은 분기모듈론리를 조종하며 나머지비트들은 주소를 제공한다. 첫번째 형식화에서 다음주소는 다음순차주소라든가 명령등록기로부터 나온 주소이다. 두번째 형식화에서 조건분기나 무조건분기가 결정된다. 이 방식의 한가지 결함은 개개의 분기명령실행에 옹근 하나의 주기가 소비되는것이다. 다른 방식들에서 주소발생은 조종기억호출을 최소화하기 위하여 조종신호발생과 같은 주기기간에 하게 한다.

우에서 서술된 방법들은 일반적인데 특정한 실현은 이 기술들의 변종이든가 조합에 의하여 이루어 진다.

3. 주소발생

우리는 형식화에 대한 고찰과 일반적론리요구의 견지에서 순서화문제를 고찰하였다. 다음주소가 어디에서 어떻게 만들어 지는가 하는것을 다른 견지에서 보기로 하자.

표 15-3 에 여러가지 주소발생방법들을 주었다. 이것들은 주소가 마이크로명령에서 명백히 유용한 명시적인 기술들과 주소를 발생하기 위한 보충적인 론리를 요구하는 암시적기술로 나눌수 있다.

표 15-3. 마이크로명령의 주소발생기술

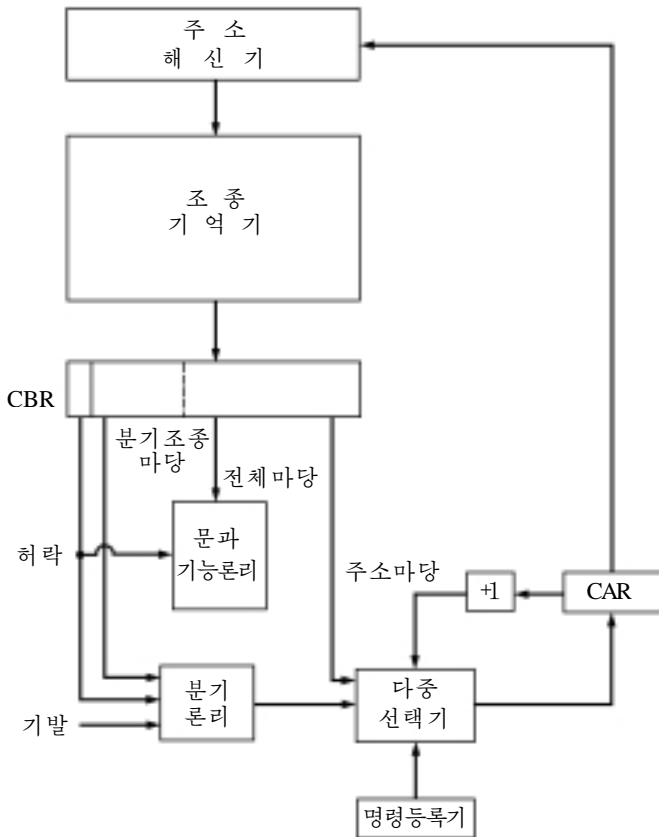
명시적	암시적
두마당	배치
무조건분기	추가
조건분기	오차조종

여기서는 주로 명시적기술을 취급하였다. 두마당방식을 리용하여 두개의 서로 다른 주소를 개개의 마이크로명령과 함께 리용한다. 하나의 주소마당이라든가 가변형식화를 리용하여 여러가지 분기명령을 실현할수 있다. 조건분기명령은 다음과 같은 정보의 형태에 의존한다.

- ALU 기발들
- 조작코드부분이든가 기계명령의 주소방식마당
- 부호비트와 같은 선택된 등록기부분
- 조종장치안의 상태비트

그림 15-8. 분기조종론리, 가변형식

여러가지의 암시적기술이 또한 자주 리용된다. 이들가운데서 하나 즉 배치화가 모든 설계를 가상화할것을 요구한다. 기계명령의 조작코드부분은 마이크로명령주소에 배치되여야 한다. 이것은 명령주기당 오직 한번만 제기된다.

일반암시적기술은 완료주소를 만들기 위하여 주소의 두 부분을 결합하거나 더하는 방법이다. 이 방식은 IBM/360 계렬[TUCK67]에서가 아니라 S/370 모델의 대부분에서 리용되였다. 여기서는 IBM 3033을 실례로 실명하려고 한다.

IBM 3033의 조종주소등록기는 13bit 길이를 가지며 그림 15-9에 보여 주었다. 주소의 두 부분은 구별할수 있다. 제일 웃자리 8bit(00～07)는 한 마이크로명령주기로부터 다음마이크로명령주기까지 일반적으로 변하지 않는다. 마이크로명령이 실행되는 동안 이 8bit는 마이크로명령의 8bit 마당(BA 마당)으로부터 조종주소등록기의 제일 웃자리 8bit에로 직접 넘어 간다. 조종기억기안에서 32개의 마이크로명령을 하나의 블로크로 정의한다. 조종주소등록기의 나머지 5bit는 다음에 꺼내야 할 마이크로명령의 정해 진 주소를 규정

하는 부분이다. 매개 비트는 현행마이크로명령안의 4bit 마당(7bit 마당을 제외)에 의하여 결정된다. 이 마당은 대응하는 비트를 설정하기 위한 조건을 규정한것이다. 례하면 조종등록기안의 비트가 ALU 연산의 마지막에 자리올림이 생기는가 안생기는가에 따라 1 혹은 0 으로 설정될수 있다.

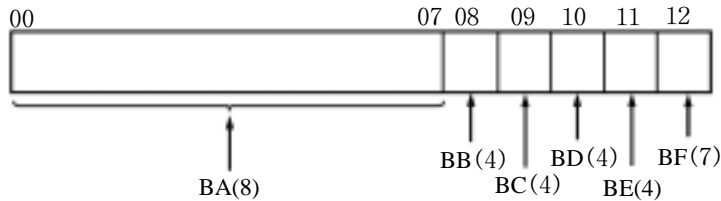



그림 15-9. IBM 3033 조종주소등록기

마지막방식을 표 15-3 에 주었는데 이것을 나머지조종이라고 한다. 이 방식은 조종장치안의 림시등록기에 이미 보관된 마이크로명령주소의 리용을 포괄하고 있다. 례하면 일부 마이크로명령들은 보조프로그람처리기술을 장비하고 있다. 내부등록기든가 등록기의 탄창은 복귀주소를 보관하군 한다. 이 방식의 실례를 LSI-11 에서 리용하였다. 그에 대하여 보기로 하자.

4. LSI-11 마이크로명령순서화

LSI-11 은 PDP-11 를 극소형콤퓨터화한것으로서 체계의 기본요소들은 한 기판에 조립하였다. LSI-11 은 마이크로프로그람조종장치를 리용한것이다[SEBE76].

LSI-11 은 22bit 의 마이크로명령과 2K 22bit 단어의 조종기억기가 있다. 다른 마이크로명령주소는 다섯가지 방법가운데서 어느 하나에 의하여 결정된다.

- **다음순차주소:** 다른 명령이 없는 경우에 조종장치안의 조종주소등록기는 하나만큼 증가된다.
- **조작코드배치:** 매개 명령주기의 시작에서 다음에 실행할 마이크로명령의 주소는 조작코드에 의하여 결정된다.
- **보조프로그람호출기능:** 앞에서 설명하였다.
- **새치기검사:** 어떤 마이크로명령들은 새치기에 대한 검사를 규정한다. 만일 새치기가 일어 난 경우 이것이 다음에 실행할 마이크로명령의 주소를 결정한다.
- **분기:** 조건 및 무조건분기마이크로명령이 리용된다.

한 준위보조프로그람호출기능이 제공되여 있다. 매개 마이크로명령안에서 1bit 가 이 과제를 지적한다. 이 비트가 설정되면 11bit 로 된 복귀를 규정하고 보조마이크로명령은 복귀등록기로부터 조종주소등록기에 넣어 지게 한다.

복귀는 무조건분기명령의 한 형태이다. 무조건분기의 다른 형태는 조종주소등록기의 비트들이 11bit 의 마이크로명령으로부터 넣어 지게 한다. 조건분기명령은 마이크로명령안의 4bit 의 검사코드를 리용한다. 이 코드들은 분기를 결정하는 여러가지 ALU 조건코드의 검사를 규정한다. 만일 조건이 참이 아니면 다음순서의 주소를 선택한다. 참인 경우 조종주소등록기의 제일 아래자리 8bit 에 마이크로명령의 8bit 가 넣어 진다. 이것은 분기가 256 단어페지의 기억기내에서 진행될수 있게 한다.

알고 있는바와 같이 LSI-11 은 조종장치내에 강력한 주소순서화기구가 장비되여 있다. 이것으로 하여 마이크로프로그람작성자들은 고찰에서 유연성을 보장할수 있고 마이크로프로그람작성을 쉽게 할수 있다. 다른 한편 이 방식은 보다 더 많은 조종장치론리를 요구한다.

제 3 절. 마이크로명령실행

마이크로명령주기는 마이크로프로그람화된 처리장치에서 기본문제로 된다. 매개 명령주기는 두개의 부분으로 되여 있는데 그것은 꺼내기와 실행주기이다. 꺼내기부분은 마이크로주소의 생성에 의하여 결정되는데 이에 대하여서는 앞절에서 취급하였다. 이 절에서도 마이크로명령에 대하여 취급한다.

마이크로명령의 실행효과는 조종신호를 발생시키는것이라는것을 상기해 보자. 이 신호의 일부는 처리장치의 내부를 조종한다.나머지신호들은 외부조종모선이든가 다른 외부대면부를 조종하는 신호이다. 흔히 있는 기능을 리용하여 다음마이크로명령의 주소를 결정한다.

앞에서 진행한 서술에서는 그림 15-10 에서 보여 준 조종장치의 구성을 론의하였다. 그림 15-4 에서 수정된 부분을 이 절에서 강조한다. 이 그림에서 기본모듈은 지금 그려 있지 않다. 순서화론리모듈은 앞절에서 서술한 기능을 수행하기 위한 론리가 들어 있다. 이 모듈은 명령등록기, ALU 기발, 조종주소등록기(주소증가용)와 조종완충등록기를 입구로 리용하여 다음마이크로명령의 주소를 발생시킨다. 최종적으로는 실제주소와 조종비트 혹은 그것을 둘다 만들어 낸다. 이 모듈은 마이크로명령의 동기를 결정하는 박자에 의하여 조종된다.

조종론리모듈은 마이크로명령안의 일부 비트의 기능을 가지고 조종신호를 만든다. 마이크로명령의 형식화와 내용이 조종론리모듈을 복잡하게 만들것이라는것은 명백하다.

마이크로명령의 분류

마이크로명령은 여러가지 방법으로 분류할수 있다. 일반적으로 문헌에 의한 분류는 다음과 같다.

- 수직/수평
- 채워 넣기/도로빼기
- 하드/쏘프트웨어마이크로프로그람
- 직접/간접부호화

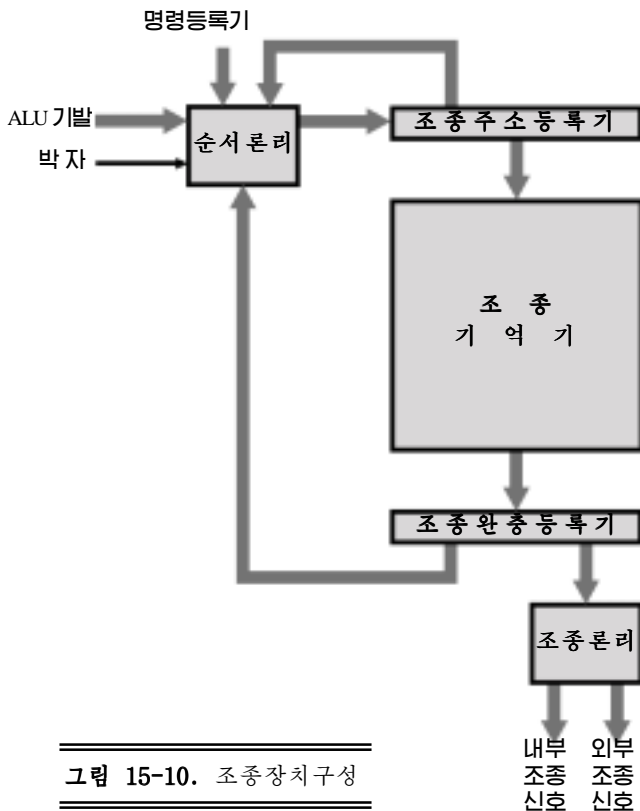



그림 15-10. 조종장치구성

이 모든것들은 마이크로명령의 형식화에 영향을 준다. 이 특성들의 쌍을 조사하는것은 마이크로명령설계의 선택을 고찰하는데 도움을 준다. 다음절에서 먼저 이 특성의 모든 쌍들에 깔려 있는 기본설계항목을 고찰하고 그다음 매개 쌍에 의하여 제기되는 개념들을 보기로 하자.

윌크스가 제기한 초기제안에서는 마이크로명령의 매개 비트는 직접 조종신호를 만들든가 혹은 직접 다음주소의 한비트를 만들었다. 우리는 이미 앞절에서 적은 마이크로명령비트를 리용하여 보다 복잡한 주소순서화를 할수 있다는것을 학습하였다. 이 방안은 보다 복잡한 순서화론리 모듈을 요구한다. 이러한 단계에 대한 류형은 조종신호와 관련된 마이크로명령부분에서도 존재한다. 조종정보를 복호화하며 그리고 조종신호를 만들기 위하여 그것을 순차로 해신하는 방법으로 조종단의 비트를 절약할수 있다.

이 부호화를 어떻게 할수 있는가? 그에 대한 대답을 하기 위하여 조종장치에 의하여 조종되는 K 개의 서로 다른 내부와 외부조종신호가 있다고 하자. 윌크스의 방안에서는

마이크로명령의 Kbit 가 모두 이 목적에 리용된다. 이것은 명령주기기간 발생되는 2^K 개의 가능한 조종신호조합을 만들수 있다. 그러나 가능한 조합이 다 리용되지 않는다면 이보다 더 잘할수 있다. 실례들에는 다음과 같은것들을 포함하고 있다:

- 두 원천주소는 같은 목적주소로 조종될수 없다(그림 14-5 에서 C_2 와 C_8).
- 하나의 등록기는 원천주소와 목적주소를 둘 다 읽을수 없다(그림 14-5 에서 C_5 와 C_{12}).
- 조종신호들가운데서 오직 하나의 패턴만이 한번에 ALU 에 가해 질수 있다.
- 조종신호들가운데서 오직 하나의 패턴만이 한번에 외부조종모선에 나타날수 있다.

그러므로 주어 진 처리장치에 대하여 조종신호의 가능한 모든 조합은 $Q<2^K$ 으로 주어 지며 그에 대한 목록을 작성할수 있다. 이것은 $(\log_2 Q)<K$ 가 성립하는 $\log_2 Q$ 비트로 해신된다. 여기에는 조종신호의 가능한 조합들을 보존하는 부호화의 모든 형식들이 집중되여 있다. 실천적으로 이러한 형태의 부호화는 쓰지 않는다. 그 리유는 다음과 같은 두가지에 있다:

- 순수한 해신(윌크스)방식처럼 실행순서를 짜기가 어렵다. 이 점은 후에 더 론의된다.
- 이것은 론리가 복잡해 지고 더우기 조종론리모듈의 속도가 떠지게 한다.

그에 대한 몇가지 절충안이 제기되고 있는데 대표적인것은 다음의 두 종류이다:

- 가능한 조합을 해신하기 위하여 필요량보다 더 많은 비트를 리용한다.
- 물리적으로 가능한 일부 조합들은 부호화할수 없다.

후자의 경우의 위태로움은 비트수를 감소시키는 효과를 가져 온다. 그러나 순리득은 $\log_2 Q$ 비트이상을 리용하는데 있다.

다음소제목에서 특별한 부호화기술을 론의한다. 이 소제목의 나머지부분에서는 부호화의 효과와 그것을 서술하는데 리용된 여러가지 용어들을 취급한다.

앞에서 진행한 고찰에 기초하면 마이크로명령형식의 조종신호부분은 하나의 스펙트르에 떨어 진다는것을 알수 있다. 한 극단에는 모든 조종신호를 위한 하나의 비트가 있으며 다른 극단에서는 높은 수준으로 부호화된 형식을 리용한다. 표 15-4 는 마이크로프로그람화된 조종장치의 다른 특성들 역시 그의 스펙트르가 대체로 부호화스펙트르의 정도에 의하여 결정되는 스펙트르대역내에 떨어 진다는것을 보여 주고 있다.

표에서 두번째 항목의 쌍은 오히려 명백하다. 순수한 윌크스방식은 많은 비트들을

요구한다. 또한 이 극단이 하드웨어의 가장 상세한 고찰을 준다는것도 명백하다. 매 조종신호는 마이크로프로그람작성자에 의하여 개별적으로 조종할수 있다. 부호화는 기능 혹은 자원을 집합하는 방법으로 수행되므로 마이크로프로그람작성자는 보다 높지만 덜 상세한 준위에서 처리장치를 고찰하고 있다. 게다가 부호화는 마이크로프로그람작성부담을 줄이도록 진행된다. 또한 모든 조종신호의 리용을 리해하고 조화롭게 종합하는 과제가 어려운 과제라는것이 명백하다. 우에서 언급한바와 같이 일반적으로 부호화의 결론들중의 하나는 어떤 다른 허용조합의 리용을 방지하는것이다.

우에서 서술한것은 마이크로프로그람작성자의 견지에서 마이크로명령설계를 론의한것이다. 그러나 부호화의 정도는 하드웨어효과로부터 고찰할수도 있다. 순수한 부호화되지 않은 형식인 경우 해신론리는 적게 혹은 전혀 요구되지 않는다. 즉 매 비트는 특별한 조종신호를 발생한다. 보다 함축되고 보다 집약된 부호화방식을 리용하므로 보다 복잡한 해신론리가 요구된다. 따라서 이것은 성능에 영향을 줄수 있다. 보다 복잡한 조종론리모듈의 문회로들을 통하여 신호를 전파시키는데 더 많은 시간이 요구된다. 따라서 부호화된 마이크로명령의 실행은 부호화되지 않은 마이크로명령의 실행보다 시간이 더 오래 걸린다.

결국 표 15-4에서 보여 준 모든 특성들은 설계전략의 범주에 속한다. 일반적으로 스

표 15-4. 마이크로명령범주

특	징
부호화하지 않는다	부호화를 해야 한다
비트가 많다	비트가 적다
하드웨어의 구체적인 고찰	하드웨어의 개괄적인 고찰
프로그람작성이 어렵다	프로그람작성이 쉽다
동시성이 충분히 활용된다	동시성을 활용하지 않는다
조종론리가 적거나 없다	조종론리가 복잡하다
실행속도가 빠르다	실행속도가 느리다
성능을 최량화할수 있다	프로그람작성을 최량화할수 있다
전 문 용 어	
비묶음	묶음
수평	수직
하드	쏘프트

펙트르의 왼쪽 끝방향으로 떨어 지는 설계는 조종장치의 성능을 최적화하는데 리용된다. 오른쪽 방향으로 떨어 지는 설계는 마이크로프로그람작성의 처리공정을 최적화하는데 크게 관계된다. 범주의 오른쪽 가까이에 있는 마이크로명령모임은 기계명령모임과 매우 류사하게 보인다. 이에 대한 좋은 실례에는 이 절에서 후에 서술되는 LSI-11 설계이다. 일반적으로 조종장치를 실현하기 위한 목적이 간단하면 그 설계는 범주의 왼쪽 끝가까이에 있을것이다. 후에 론의하겠지만 일부 체계는 여러 사용자들이 같은 마이크로명령설비를 리용하여 각이한 마이크로프로그람을 구성하도록 한다. 후자의 경우에 설계는 범주의 오른쪽 가까이에 떨어 지게 된다.

이전에 언급되였던 전문용어들에 대하여 주의를 돌리자. 표 15-4는 이 용어들의 3가지 쌍이 마이크로명령범주와 어떻게 관계되는가를 보여 주고 있다. 사실상 이 쌍들은 서로 다른 설계특성들을 강조하지 않는 같은 실체들을 서술하고 있다.

묶음정도는 주어 진 조종과제와 특별한 마이크로명령비트들사이에 동일성정도에 관계된다. 비트들이 보다 묶어 지게 되므로 주어 진 수의 비트들은 더 많은 정보를 포함한다. 결국 묶음은 부호화를 의미한다. **수평과 수직**은 마이크로명령의 상대적인 너비에 관계된다. [SIEW82]은 수직마이크로명령이 16~40bit의 범위의 길이를 가지며 수평마이크로명령이 40~100bit 범위의 길이를 가진다는것을 엄지손가락규칙으로 제기하고 있다. 하드 및 쏘프트마이크로프로그람작성은 기본조종신호와 하드웨어배치사이의 밀착성정도를 암시하는데 리용된다. 일반적으로 하드마이크로프로그람은 ROM에 고정된다. 쏘프트마이크로명령프로그람은 자주 변화시킬수 있으며 사용자의 마이크로프로그람작성을 가능하게 한다.

이 소제목을 시작하면서 언급한 술어들의 다른 쌍들은 이제 고찰하려고 하는 대상들인 직접 및 간접부호화에 귀착된다.

2. 마이크로명령부호화

실천적으로 마이크로프로그람화된 조종장치는 순수한 부호화되지 않거나 수평인 마이크로명령형식화를 리용하여 설계되지 않았다. 일부 부호화는 조종기억기의 폭을 최소한 줄이며 마이크로프로그람작성을 간단화하기 위하여 리용되였다.

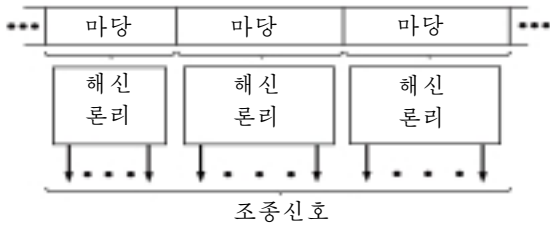
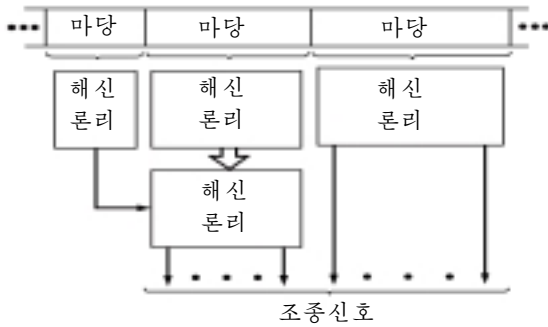
부호화의 일부 수법을 그림 15-11 ㄱ에 보여 주었다. 마이크로명령은 한조의 마당으로 구성되였다. 매개 마당에는 코드가 들어 있는데 그것은 이 코드가 해신되여 하나 혹은 여러개의 조종신호를 능동으로 만든다.

이 배렬의 의미(련관관계)에 대하여 보기로 하자. 마이크로명령이 해신될 때 매개 마당은 해신되여 조종신호를 발생시킨다. 따라서 N개의 마당을 가지고 N개의 동작을 동

시에 규정할수 있다. 매개 동작은 하나 혹은 여러개의 조종신호를 능동으로 만든다. 늘 그렇지는 않지만 일반적으로 매개 조종신호가 하나의 마당정도에 의하여 능동이 되도록 형식화를 설계하는것이 필요하다. 그런데 매개 조종신호가 최소한 하나의 마당에 의하여 능동이 될 가능성이 있는가 하는것을 명백히 해두는것이 필요하다.

그림 15-11. 마이크로 명령부호화
ㄱ-직접부호화, ㄴ-간접부호화

여기서 개별적인 마당을 고찰해 보자. Lbit로 구성된 마당에는 2L코드들 가운데서 하나가 들어 갈수 있으며 그의 매개는 서로 다른 조종신호패턴으로 부호화될수 있다. 왜냐하면 하나의 코드만이 한번에 하나의 마당에 나타날수 있기때문에 코드들은 호상 배타적이며 따라서 그들이 발생시키는 동작도 호상 배타적이다.

부호화된 마이크로명령형식의 설계는 다음과 같은 단순한 항목에서부터 시작된다.

- 형식화는 독립적인 마당들로 구성한다. 즉 매개 마당은 서로 다른 마당에 의하여 생기는 동작들이 동시에 일어 나도록 한묶음의 동작을 담고 있어야 한다.
- 마당에 의하여 규정될수 있는 선택된 동작들은 호상 배타적이 되도록 매개 마당을 정의하여야 한다. 즉 주어 진 마당에 의하여 규정된 동작들가운데서 오직 하나만 동시에 일어 날수 있다.

두 방식들은 부호화된 마이크로명령 즉 기능과 원천으로 된 마당으로 구성할수 있다. 기능부호화방식은 기계안의 기능을 규정하며 마당을 기능형식으로 표시한다. 례하면 만일 자료를 축적등록기로 전송하는데 여러가지 원천들을 리용한다면 한마당은 이 목적을 표시하며 매개 코드는 원천의 종류를 의미한다. 자원코드화는 기계가 독립적인 자원묶음을 구성하고 있는것처럼 보자는데 있으며 개개(례하면 I/O, 기억기, ALU)에 하나의 마당을 할당한다.

부호화의 다른 측면은 그것이 직접인가, 간접인가하는것이다(그림 15-11 ㄴ). 간접부호화를 리용하여 한 마당이 다른 마당에 대한 해석을 결정하군 한다. 례를 들어 8

개의 산수연산과 8개의 밀기조작을 수행할수 있는 ALU를 고찰해 보자. 1bit마당은 밀기조작인가 혹은 산수연산인가를 지적하는 마당이다. 3bit마당은 그 조작을 수행하는 마당이다. 이 방법은 일반적으로 두개의 해신준위를 가지므로 전달지연시간이 커진다.

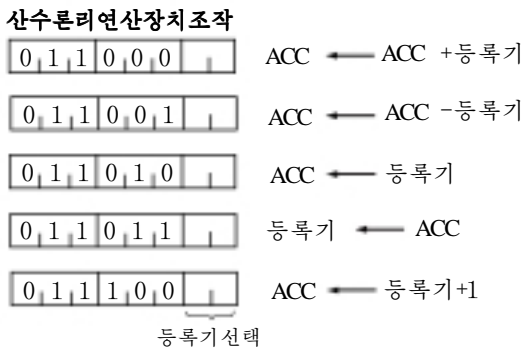
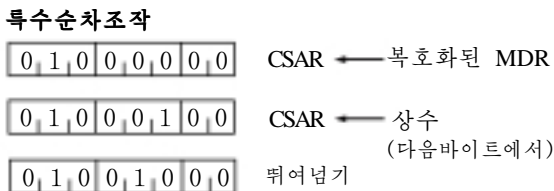
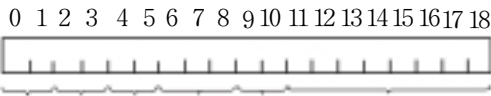



그림 15-12는 이 개념을 실례로 준것이다. 프로그람계수기와 ALU 입구를 위한 림시등록기처럼 하나의 축적등록기와 여러개의 내부등록기들을 가진 처리장치에 대하여 가정해 보자. 그림 15-12 ㄱ는 수직형식을 보여 준다. 첫 3bit는 조작의 종류를 지적하며 그다음 3bit는 조작을 부호화하며 마지막 2bit는 내부등록기를 선택한다. 그림 15-12 ㄴ는 여전히 부호화를 리용하지만 보다 더 수평방식에 접근하여 있다는것을 보여 준다. 이 경우에 서로 다른 기능이 서로 다른 마당에 나타난다.

그림 15-12. 단순한 기계에서 마이크로명령형식
ㄱ-수직마이크로명령, ㄴ-수평마이크로명령

3. LSI-11 마이크로명령실행

LSI-11은 수직마이크로명령의 좋은 실례이다. 여기서는 먼저 조종장치의 내부구성을 보고 그다음 마이크로명령형식을 보기로 하자.

LSI-11 조종장치의 구성

LSI-11은 한 기판으로 된 PDP-11 계렬의 첫 콤퓨터이다. 기판에는 3개의 LSI 소편, 마이크로명령모선(MIB)으로 된 내부모선 그리고 몇가지 대면론리회

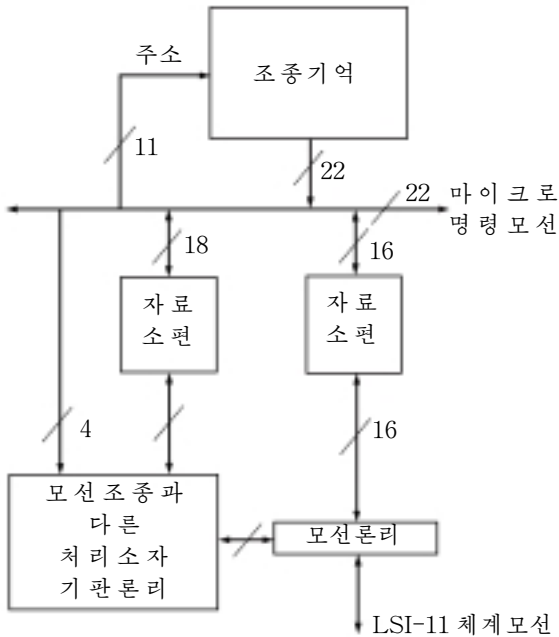



그림 15-13. LSI-11 처리소자의 간단한 구성도

로로 구성되여 있다.

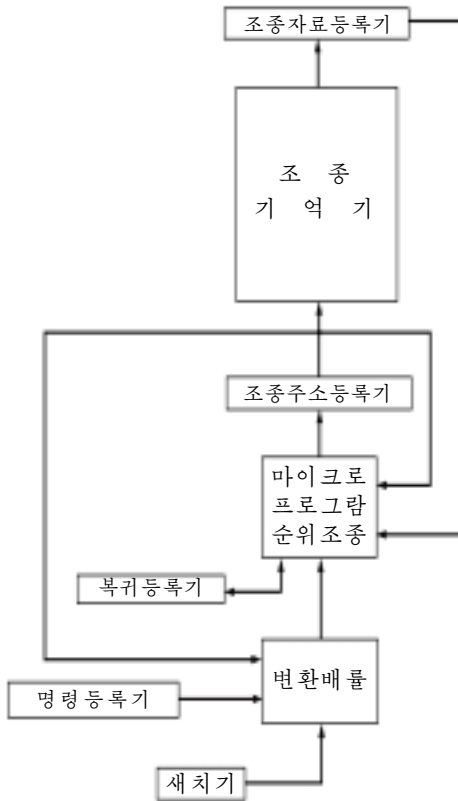
그림 15-13 에 LSI-11 처리장치의 구성에 대하여 간단히 보여주었다. 3 개의 소편은 자료소편, 조종소편, 조종기억소편을 말한다. 자료처리소편은 한개의 8bit ALU, 26 개의 8bit 등록기들, 여러개의 조건코드를 기억하기 위한 기억기로 이루어 져 있다. 26 개의 8bit 등록기들가운데서 16 개의 등록기들은 2 개씩 짝을 무어 8 개의 16bit PDP-11 의 일반등록기로 쓸수 있다. 다른 등록기들은 프로그람상태단어등록기, 기억주소등록기(MAR) 그리고 기억완충등록기로 된다. ALU 는 한번에 8bit 씩만 처리하므로 16bit PDP-11 산수연산을 실현하려면 ALU 를 통하여 8bit 자료를 두번 통과시켜야 한다. 이 과정은 마이크로프로그람에 의하여 조종된다.

조종기억소편이든가 소편들은 22bit 폭의 조종기억기로 구성되여 있다. 조종소편은 마이크로명령순서화와 실행을 위한 론리를 수행한다. 거기에는 조종주소등록기와 조종자료등록기 그리고 기계어명령등록기의 내용이 들어 있다.

모든 요소들은 다 MIB 를 통하여 결합된다. 마이크로명령을 꺼내는 기간 조종소편은 MIB 에 11bit 의 주소를 내보낸다. 조종기억기가 호출되면서 MIB 에 놓여 있는 22bit 로 된 마이크로명령이 출력된다. 이가운데서 아래자리 16bit 는 자료소편으로, 한편 아래자리 18bit 는 조종소편으로 출력된다. 웃자리 4bit 는 특수한 처리장치기능들을 조종한다.

그림 15-14 는 LSI-11 조종장치에 대하여 간단하면서도 보다 구체적으로 고찰할수 있게 하였다. 그림에서는 개별적인 소편의 경계는 무시하였다. 제 2 절에서 서술된 주소순서화는 두개 모듈로 구성되여 있다. 모든 순서조종은 마이크로순서조종모듈에 의하여 조종되는데 이 모듈은 마이크로명령주소등록기의 증가와 무조건분기의 실현을 할수 있다. 주소계산의 다른 형태는 개별적인 주소변환배렬에 의하여 수행된다. 이것은 마이크로명령, 기계어명령, 마이크로명령프로그람계수기, 새치기등록기에 기초하여 주소를 할당시키는 조합회로이다.

그림 15-14. LSI-11 구성

주소변환배렬은 다음과 같은 경우에 리용된다.

- 조작코드는 마이크로루틴(보조프로그람)의 시작을 결정하는데 리용된다.
- 필요한 때에 마이크로명령의 주소방식비트들은 해당한 주소화를 수행하는것을 검사하게 된다.
- 새치기조건들이 주기적으로 검사된다.
- 조건분기마이크로명령을 평가한다.

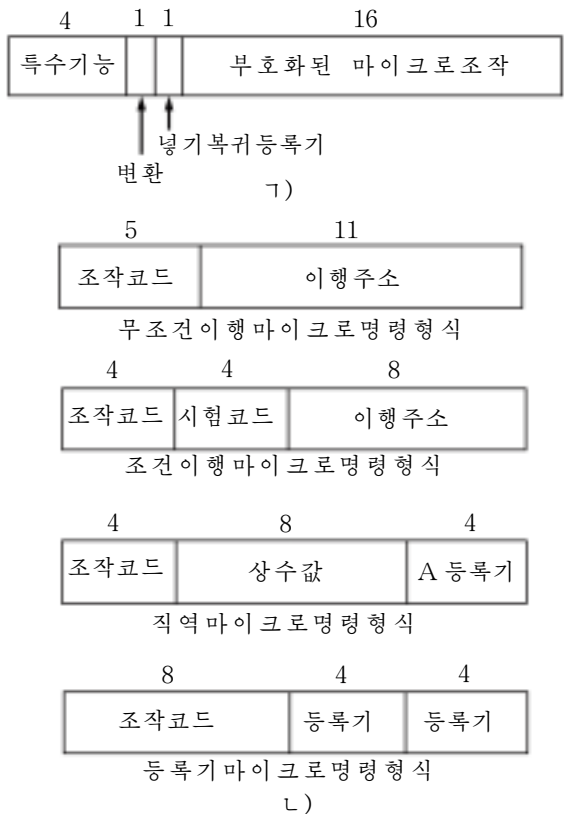



그림 15-15. LSI-11 마이크로명령형식
ㄱ-완전 LSI-11 마이크로명령형식, ㄴ-LSI-11 마이크로명령의 부호화된 부분의 형식

LSI-11 마이크로명령형식

LSI-11 는 22bit 폭을 가진 수직마이크로명령형식을 리용한다. 마이크로명령은 그것을 실현한 PDP-11 기계어명령과 매우 류사하다. 이 설계는 수직프로그람설계하에서 조종장치의 성능을 최적화하자는데 있다. 표 15-5는 일부 LSI-11마이크로명령을 보여주었다.

그림 15-15 는 22bit LSI-11 마이크로명령형식을 보여 주었다. 웃자리 4bit 는 처리장치의 특수한 기능을 조종한다. 변환비트는 주소변환배렬이 새치기를 검사하게 한다. 넘기

복귀등록기비트는 마이크로명령의 끝에서 리용하여 다음마이크로명령주소와 복귀등록기로부터 읽기 되게 한다.

나머지 16bit 는 부호화된 마이크로명령을 위하여 리용된다. 이 마이크로명령형식은 가변길이조작코드와 하나 혹은 여러개의 연산수를 가진 기계명령과 매우 류사하다.

표 15-5. LSI-11 의 일부 마이크로명령

산수연산	일반조작
단어더하기(바이트, 문자)	단어(바이트)이동
단어검사(바이트, 문자)	이행
한단어(바이트) 증가	되돌이
두단어(바이트) 증가	조건이행
단어(바이트)부정 취하기	기발설정(재설정)
조건에 따라 바이트 증가(감소)	G 에 0 을 넣기
조건에 따라 단어(바이트)더하기	조건에 따라 단어(바이트)이동
자리올림을 포함한 단어(바이트)더하기	**입출구조작**
조건에 따라 10 진수더하기	단어(바이트)입구
단어(바이트)덜기	상태단어(바이트)입구
단어(바이트)하나 감소	읽기
론리조작	쓰기
단어(바이트, 문자)론리곱하기	읽기(쓰기)한 다음 한단어(바이트) 증가
단어(바이트)검사	읽기(쓰기)한 다음 두단어(바이트) 증가
단어(바이트)론리더하기	읽기(쓰기)응답
단어(바이트)배타적론리합	단어출구(바이트, 상태)
단어(바이트)비트지우기	
자리올림이 있는(없는) 단어(바이트)오른쪽(왼쪽)밀기	
단어(바이트)보수	

4. IBM 3033 마이크로명령실행

표준 IBM 3033 조종기억기는 4K 단어로 구성되여 있다. 첫번째의 절반인 0000～07FF 에는 108bit 의 마이크로명령이 들어 있으며 나머지 0800～0FFF 는 126bit 의 마이크로명령이 기억된다. 이 형식을 그림 15-16 에 주었다. 이것은 어느 정도 수평형식이지만 부호화는 쉽다. 이 형식화의 기본마당을 표 15-6 에 종합하여 주었다.

ALU 는 사용자가 볼수 없는 4 개의 전용등록기 A, B, C, D 로부터 나오는 입구자료를 처리한다. 마이크로명령형식은 사용자등록기로부터 나오는 자료를 이 등록기에 넣기 위한 마당들로 구성되여 있다. 이 사용자가 쓸수 있는 등록기(사용자용등록기)는 ALU 기능을 수행하거나 결과를 기억하기 위하여 사용자가 쓸수 있는 등록기를 규정한다. 이외에 등록기와 기억기사이에 자료를 넣거나 기억시키기 위한 마당이 있다.

IBM 3033 의 순서화과정을 제 2 절에서 취급하였다.

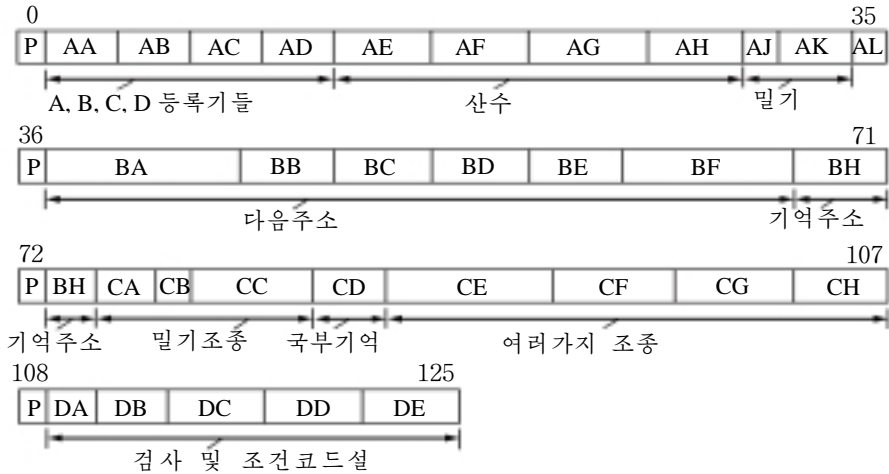



그림 15-16. IBM 3033 마이크로명령형식

표 15-6. IBM 3033 마이크로명령조종마당

ALU 조종마당	
AA(3)	임의의 자료등록기에서 A 등록기에 넣기
AB(3)	임의의 자료등록기에서 B 등록기에 넣기
AC(3)	임의의 자료등록기에서 C 등록기에 넣기
AD(3)	임의의 자료등록기에서 D 등록기에 넣기
AE(4)	규정된 A 비트를 ALU 에 보내기
AF(4)	규정된 B 비트를 ALU 에 보내기
AG(5)	A 입구에 대한 ALU 론리연산을 규정
AH(4)	B 입구에 대한 ALU 론리연산을 규정
AJ(1)	B 쪽에 있는 ALU 에 D 혹은 B 입구를 규정
AK(4)	밀기등록기에 산수연산출구 보내기
CB(1)	밀기등록기 능동
CC(5)	론리적 및 자리올림기능을 규정
CE(7)	밀기등록기값을 규정
CA(3)	F 등록기에 적재
순서화와 분기마당	
AL(1)	조작완료와 분기수행
BA(8)	조종주소등록기의 높은 자리비트(00-07)를 설정
BB(4)	조종주소등록기의 비트 8 을 설정하기 위해 조건을 규정
BC(4)	조종주소등록기의 비트 9 을 설정하기 위해 조건을 규정
BD(4)	조종주소등록기의 비트 10 을 설정하기 위해 조건을 규정
BE(4)	조종주소등록기의 비트 11 을 설정하기 위해 조건을 규정
BF(4)	조종주소등록기의 비트 12 을 설정하기 위해 조건을 규정

제 4 절. TI 8800

텍사스 인스트르먼츠회사（TI:Texas Instruments）가 개발한 일명 8800 쏘프트웨어 개발기판(SDB)은 마이크로프로그람가능한 32bit 콤퓨터카드이다. 이 체계에는 ROM 이라기 보다 RAM 에 실현하는 쓰기조종기억기를 가지고 있다. ROM 조종기억기를 가진 그러한 체계는 마이크로프로그람가능한 체계의 속도나 밀집도를 달성할수 없었다. 그러나 이것은 개발용모형이라든가 교육용목적에는 쓸수 있다.

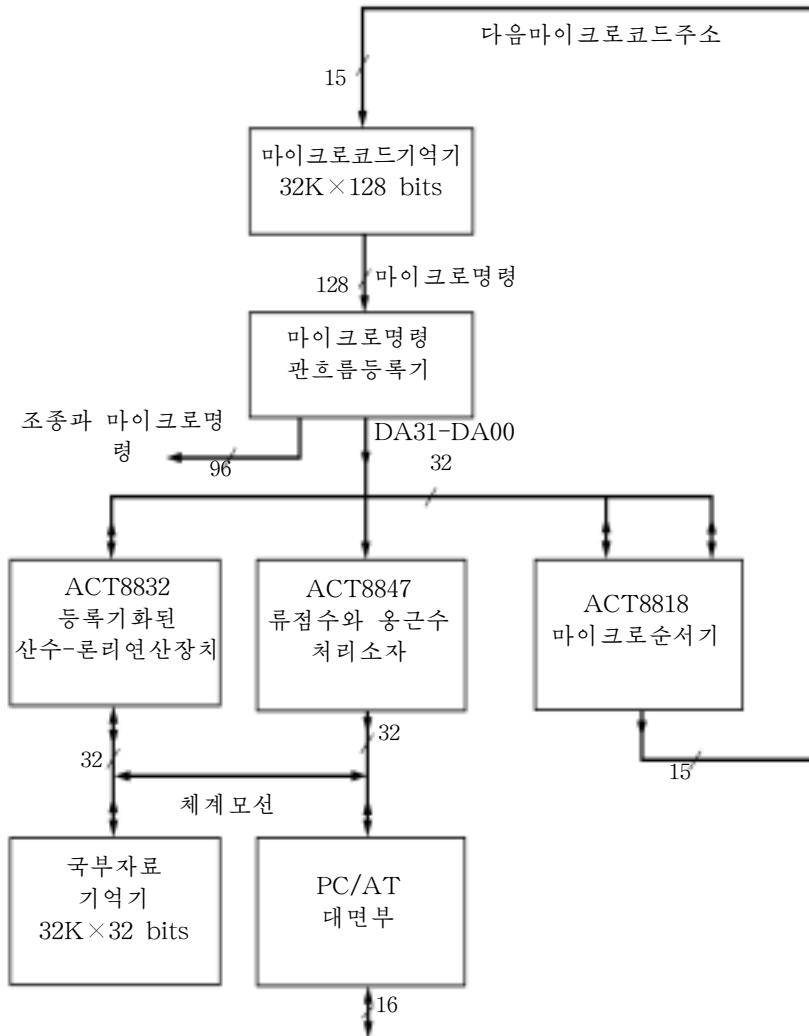



그림 15-17. TI 8800 불로크도

8800 SDB는 다음과 같은 구성요소로 되여 있다(그림 15-17).

- 마이크로코드기억기
- 마이크로순서기
- 32bit 산수-론리연산장치
- 류점수 및 옹근수처리소자
- 국부자료기억기

체계의 내부요소들은 두개의 모선에 련결되여 있다. DA 모선은 마이크로명령자료마당으로부터 ALU, 류점수처리장치 혹은 마이크로순서기에로 자료를 전송하기 위한것이다. 후자의 경우에 자료는 분기명령의 경우에 리용되는 주소로 구성되여 있다. 모선은 또한 다른 요소에 자료를 전송하기 위하여 ALU 혹은 마이크로순서기에 리용된다. 체계 Y 모선은 ALU와 류점수처리장치를 PC대면부를 통하여 국부기억기와 외부모듈에 련결한다.

이 기판은 IBM-PC 호환가능한 주콤퓨터에 적합하다. 주콤퓨터는 마이크로코드번역과 오유수정을 위한 가동환경이다.

1. 마이크로명령형식

8800을 위한 마이크로명령형식은 128bit로 구성되여 있는데 표 15-7에서 보여 준것처럼 기능마당으로 나뉘여 져 있다. 매개 마당은 하나 혹은 여러개의 비트들로 되여 있으며 5개의 기본부류로 구성되여 있다:

- 기판의 조종
- 8847 류점수 및 옹근수처리장치 소편
- 8832 등록기화된 산수-론리연산장치(ALU)
- 8818 마이크로순서기
- WCS 자료마당

그림 15-17에서 보여 주는바와 같이 WCS자료마당의 32bit는 DA모선에 출력되며 그것은 자료로서 ALU 류점수처리장치 혹은 마이크로순서기에 공급(입력)된다. 마이크로명령의 다른 96개 비트(마당 1-27)는 해당한 모듈에 직접 가해 지는 조종신호로 된다. 설명을 간단히 하기 위하여 이외의 다른 련결관계는 그림 15-17에서 보여 주지 않았다.

첫 6개 마당은 개별적요소에 대한 조종이라기보다 기판의 조종에 대한 조작을 취급한다.

표 15-7. TI 8800 마이크로명령형식

마당번호	비트수	해 설
		기판의 조종
1	5	조건코드입력선택
2	1	외부적인 I/O 요구신호 허락/금지
3	2	국부자료읽기/쓰기조작 허락/금지
4	1	넣기상태/넣기하지않은 상태
5	2	Y 모선을 구동하는 장치선택
6	2	DA 모선을 구동하는 장치선택
		8847 류점수 및 옹근수처리장치소편
7	1	C 등록기조종: 박자, 박자없음
8	1	Y 모선에 대한 높은자리비트 혹은 낮은자리비트선택
9	1	C 등록기자료원천: ALU 와 다중선택기
10	4	ALU 와 MUL 에 대하여 IEEE 및 FAST 방식선택
11	8	자료연산수에 대한 원천선택: RA 등록기, RB 등록기, P 등록기, S 등록기, C 등록기
12	1	RB 등록기조종: 박자, 박자 없음
13	1	RA 등록기조종: 박자, 박자 없음
14	2	자료원천확증
15	2	관흐름등록기 허락/금지
16	11	8847 ALU 기능
		8832 등록기를 가진 산수-론리연산장치(ALU)
17	2	선택된 등록기로 허락/금지된 자료출력, 쓰기: 높은자리비트절반 낮은자리비트절반
18	2	등록기파일자료원천 선택: DA 모선, DB 모선, ALU Y MUX 출구, 체계 Y 모선
19	3	밀기명령변경자
20	1	자리올림을 하겠는가, 안하겠는가
21	2	ALU 방식설정: 32, 16 혹은 8bit
22	2	다중선택기 S 에 대한 입구선택: 등록기 파일, DB 모선, MQ 등록기
23	1	다중선택기 R 에 대한 입구 선택: 등록기파일 DA 모선
24	6	쓰기하려는 파일 C 에서 등록기선택
25	6	읽기하려는 파일 B 에서 등록기선택
26	6	쓰기하려는 파일 A 에서 등록기선택
27	8	ALU 기능
		8818 마이크로순서기
28	12	8818 입구신호조종

마당번호	비트수	해 설
		WCS 자료마당
29	16	쓰기 가능한 조종기억자료마당의 높은자리비트
30	16	쓰기 가능한 조종기억자료마당의 낮은자리비트

조종조작에는 다음과 같은것이 포함되여 있다:

- 순서기조종을 위한 조작코드의 선택. 마당1의 첫 비트는 조건기발이 1 혹은 0으로 설정되였는가를 지적하며 나머지 4bit는 어느 기발이 설정되였는가를 지적한다.
- PC/AT에 I/O 요구를 보내기
- 국부자료기억기에 대한 읽기/쓰기조작
- 체계 Y 모선을 구동하는 장치의 결정. 모선에 련결된 4개 장치가운데서 하나를 선택한다(그림 15-17).

마지막32bit는 자료마당이며 거기에는 부분적인 마이크로명령에 규정된 정보가 들어있다.

마이크로명령의 나머지마당들은 그것이 조종하는 장치의 경우에 자주 론의된다. 이 절의 마지막에 마이크로순서기와 등록기화, ALU를 고찰한다. 류점수장치는 새로운 개념이 아니므로 취급하지 않고 넘어 간다.

2. 마이크로순서기

8818의 기본기능은 마이크로프로그람을 위하여 다음마이크로명령주소를 생성하는것이다. 이 15bit의 주소는 마이크로코드기억기에 가해진다(그림 15-17).

그다음의 주소는 다섯개의 원천가운데서 어느 하나로부터 선택된다:

1. 마이크로프로그람계수(MPC)등록기. 이것은 같은 주소를 다시 리용하기 위하여 반복하거나 다음명령을 실행하기 위하여 주소를 하나만큼 증가시키는데 리용된다.
2. 탄창. 이것은 반복순환이든가 새치기로부터의 복귀와 같은 마이크로보조루틴호출을 지원한다.
3. DRA, DRB 포구들. 이 포구들은 마이크로프로그람을 발생시키는 외부하드웨어로부터 두개의 보충적인 통로를 제공한다. 이 두개 포구는 DA 모선의 제일 웃자리와 아래자리 16bit에 각각 련결되여 있다. 이것은 마이크로순서기가 현행마이크로명령의 WCS 자료마당이든가 ALU에 의하여 계산되는 결과로부터 나오는 다음명령주소를 얻자는데 있다.
4. 등록기계수기 RCA와 RCB. 이 계수들은 주소기억기를 보충하는데 리용된다.
5. 외부새치기를 지원하기 위하여 쌍방향 Y 포구에 대한 외부입구

그림 15-18은 8818의 론리블로크도를 보여 준다. 이 소자는 다음과 같은 기본기능블

로크로 이루어 져 있다.

- 등록기와 주소증가기(+1 증가)로 구성된 16bit 마이크로프로그람계수기(MPC)
- 두개의 등록기계수기. 이 등록기계수기는 순환과 반복을 계수하여 분기주소를 기억하며 외부장치를 조종한다.

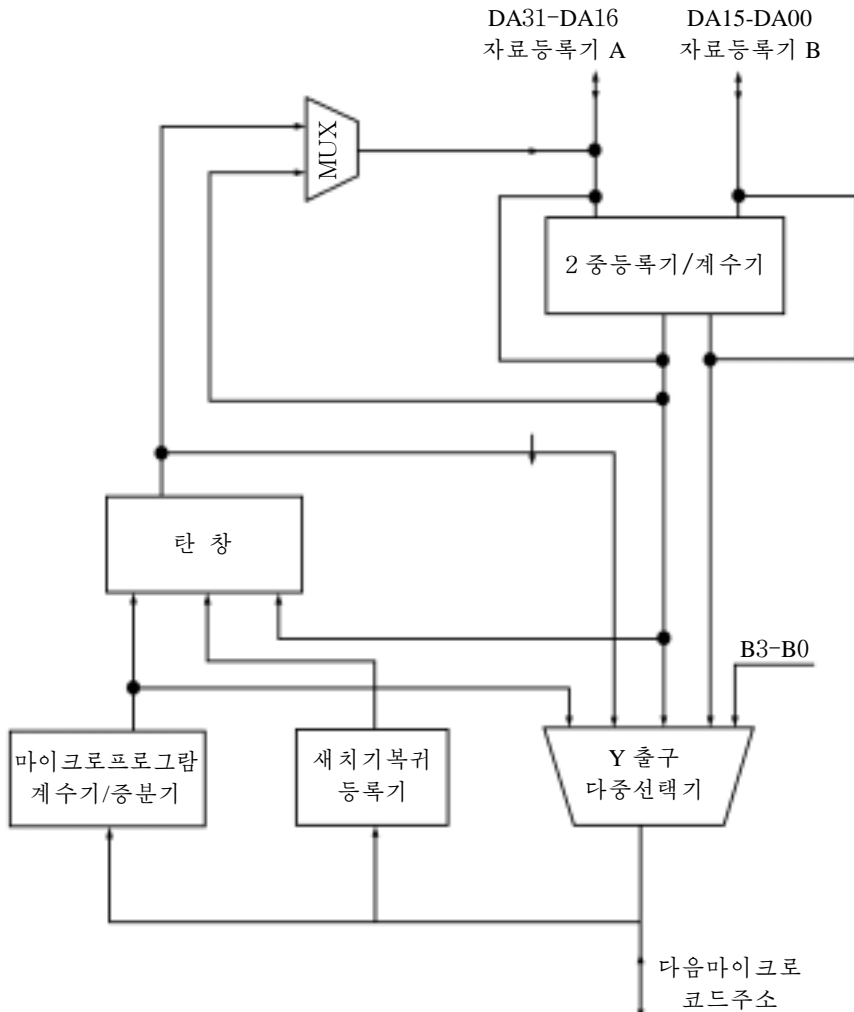



그림 15-18. TI 8818 마이크로순서기

- 16bit × 65 단어탄창. 이 탄창은 마이크로프로그람 보조루틴모듈과 새치기에 리용한다.
- 새치기복귀등록기와 Y 출구. 이것들은 마이크로명령준위에서 새치기처리를 하기 위한것이다.

- 다음주소가 MPC, RCA, RCB, 외부모선 DRA 와 DRB 와 탄창으로부터 선택될수 있는 Y 출구다중선택기

등록기들과 계수기들

등록기 RCA 와 RCB 는 DA 모선, 현행마이크로명령이든가 ALU 출구로부터 나오는 값이 넣어 진다. 그 값은 실행시 흐름을 조종하는 계수기값으로서 리용되거나 호출될 때마다 자동적으로 하나 감소된다. 이 값은 또한 Y 출구다중선택기에 공급되는 마이크로명령주소로서 리용된다. 하나의 마이크로명령주기기간 두 등록기에 대한 개별적조종은 두 등록기에 대한 동시적인 감소를 제외하고 다 할수 있다.

탄창

탄창은 여러 준위의 네스트호출이든가 새치기 그리고 분기와 순환을 지원한다. 이 조작은 모든 처리장치가 아니라 조종장치를 말하는것이며 가지고 있는 주소는 조종기억기안의 마이크로명령의 주소이다.

탄창조작은 다음과 같은 6 가지가 가능하다:

1. 지우기. 탄창지시기를 령으로 설정하고 탄창의 내용을 비운다.
2. 탄창에서 꺼내기. 이것은 탄창지시기를 하나 던다.
3. 탄창에 넣기. MPC, 새치기복귀등록기 혹은 DRA 모선의 내용을 탄창에 넣고 탄창지시기를 하나 증가시킨다.
4. 읽기. 이것은 주소가 Y 출구다중선택기에서 가능한 읽기지시기에 의하여 지적되게 한다.
5. 유지. 탄창지시기의 주소가 변하지 않고 남아 있게 한다.
6. 탄창지시기에 입력. 이것은 DRA 의 아래자리 6bit 를 탄창지시기에 입력한다.

마이크로순서기의 조종

마이크로순서기는 현행마이크로명령의 12bit 마당에 의하여 먼저 조종된다. 이 마당은 다음과 같은 보조마당으로 되여 있다.

- **OSEL(1bit):** 출력선택. 이것은 값이 DRA 모선에 넣어 지는 다중선택기의 출구에 나타나게 한다. 출력은 탄창이든가 등록기 RCA 로부터 나오도록 선택한다. 그때 DRA 는 Y 출구 다중선택기들과 등록기 RCA 에 대한 입력으로서 복사한다.
- **SELDR(1bit):** DR 모선선택. 이것이 만일 1 로 선택되면 이 비트는 외부 DA 모선을 DRA/DRB 에 대한 입력으로서 선택한다. 0 으로 선택하면 DRA 다중선택기의 출구를 DRA 모선(OSEL 에 의하여 조종됨)으로 선택하여 RCB 의 내용을 DRB 모

선으로 출력한다.

- **ZEROIN(1bit):** 조건분기를 지적하는데 리용된다. 그다음 마이크로순서기의 동작은 마당 1에서 선택된 조건코드에 의존하게 된다 (표 15-7).
- **RC2-RC0(3bit):** 등록기조종. 이 비트들은 등록기 RCA와 RCB의 내용에서의 변화를 결정한다. 매개 등록기는 값의 변화없이 그대로 가지고 있던가 하나 덜던가 DRA/DRB 모선으로부터 읽기를 할수 있다.
- **S2-S0(3bit):** 탄창조종. 이 비트들은 탄창조작이 진행되고 있음을 결정한다.
- **MUX2-MUX0:** 출력조종. 이 비트들은 조건코드와 함께 Y출구다중선택기와 다음 마이크로명령주소를 조종한다.

다중선택기는 탄창, DRA, DRB 그리고 MPC로부터 나오는 출력을 조종한다. 이 비트들은 프로그람작성자가 개별적으로 설정할수 있다. 그러나 꼭 이렇게 하는것은 아니다. 오히려 프로그람작성자는 일반적으로 쓰이는 비트패턴(기계어)과 같은 아쎔블리언어를 리용한다. 표 15-8은 마당 28에 대한 15개의 아쎔블리언어를 표로 주었다. 마이크로코드번역프로그람은 이것을 해당한 비트패턴으로 번역한다.

례하면 명령 INC88181은 현재 선택된 조건코드가 1이면 다음마이크로명령이 순서대로 선택되게 하는데 쓰인다.

표 15-8로부터

INC88181=000000111110

을 얻을수 있다. 이것을 다음과 같이 직접 해신할수 있다.

표 15-8. TI 8818 마이크로순서기의 마이크로명령비트(마당 28)

기억부호	값	해설
RST8818	000000000110	재설정명령
BRA88181	011000111000	DRA명령에로의 분기
BRA88180	010000111110	DRA명령에로의 분기
INC88181	000000111110	계속명령
INC88180	001000001000	계속명령
CAL88181	010000110000	DRA에 의하여 규정된 주소에서 보조루틴으로 뛰여넘기
CAL88180	010000101110	DRA에 의하여 규정된 주소에서 보조루틴으로 뛰여넘기
RET8818	000000011010	보조루틴으로부터 되돌이

표계속

기억부호	값	해 설
PUSH8818	000000110111	탄창에 새치기되돌이주소의 밀어넣기
POP8818	100000010000	새치기로부터 되돌이
LOADDRA	000010111110	DA 모선으로부터 DRA 계수기적재
LOADDRB	000110111110	DA 모선으로부터 DRB 계수기적재
LOADDRAB	000110111100	DRA/DRB 적재
DECRDRA	010001111100	령이면 DRA 계수기와 분기감소
DECRDRB	010101111100	령이면 DRB 계수기와 분기감소

- **OSEL＝0:** RCA 를 DRA 출구 MUX 로부터 나오는 출구로서 선택한다. 이 경우에 선택은 적합하지 않다.
- **SELCR＝0:** 앞에서 규정된것처럼 이것은 이 명령에 대하여 적합치 않다.
- **ZEROIN＝0:** MUX 의 값과 결합되여 분기가 취해지지 않았음을 지적한다.
- **R＝000:** RA 와 RC 의 현재값을 유지한다.
- **S＝111:** 탄창의 현재값을 유지한다.
- **MUX＝110:** 조건코드가 1 이면 MPC 를 선택하고 조건코드가 0 이면 DRA 를 선택한다.

3. 등록기화된 ALU

8832 는 64 개의 등록기를 가진 32bit ALU 로서 4 개의 8bit ALU, 두개의 16bit ALU, 그리고 하나의 32bit ALU 로서 동작하도록 구성되여 있다.

8832 는 마이크로명령마당 17～27 에 의하여 만들어 지는 39bit 에 의하여 조종된다(표 15-7). 이것은 조종신호로서 ALU 에 가해 진다. 더우기 그림 15-17 에서 지적된것처럼 8832 는 32bit DA 모선과 32bit 체계 Y 모선과 외부련결된다. DA 로부터 나오는 입력은 64word 등록기파일과 ALU 론리모듈에 입력자료로서 동시에 가해 진다. ALU 연산과 밀기의 결과는 출력모선이든가 체계 Y 모선에 출력된다. 결과는 또한 내부등록기파일에 다시 입력된다.

3 개의 6bit 주소포구는 두개의 연산수꺼내기와 하나의 연산수쓰기가 등록기파일내에서 동시에 수행되도록 한다. MQ 밀기회로와 MQ 등록기는 또한 8bit, 16bit 의 배정확도 조작을 실현하기 위하여 독립적으로 기능을 수행할수 있게 구성되여 있다.

매개 마이크로명령의 마당 17～26 은 자료가 8832 와 외부환경사이에서 움직이는 방

법을 조종한다. 그 마당은 다음과 같다:

17. 쓰기허락. 이 두비트는 32bit에 대한 쓰기를 하는가, 웃자리 16bit 혹은 아래자리 16bit 를 따로따로 쓰기하는가와 등록기파일에 쓰기를 하지 않는가 하는것을 규정한다.

18. 등록기파일자료원천선택. 만일 등록기파일에 대한 쓰기가 생기면 이 두비트는 원천지인 DA모선, DB모선, ALU출구 그리고 체계 Y모선가운데서 어느 하나를 선택한다.

19. 밀기명령갱신. 밀기명령이 진행되는 기간 끝을 채우는 비트들의 제공과 밀기된 비트들의 읽기와 관련된 선택을 규정한다.

20. 자리올림입구. 이 비트로 연산을 하는 동안 ALU에 자리올림을 하겠는가를 지적한다.

21. ALU기본방식. 8832는 하나의 32bit ALU 혹은 4개의 8bit ALU로 동작하게 구성할수 있다.

22. S입구. ALU론리모듈입력은 S와 R 다중선택기라고 하는 두개의 내부 다중선택기로부터 받는다. 이 마당은 S다중선택기에 의하여 제공되는 입구인 등록기파일, DB 모선, MQ 등록기가운데서 하나를 선택한다. 원천등록기는 마당 25에 의하여 규정된다.

23. R 입구. R 다중선택기에 의하여 제공되는 입구인 등록기파일이든가 DA 모선가운데서 어느 하나를 선택한다.

24. 목적등록기. 목적연산수를 위하여 리용되는 등록기파일에 대한 주소이다.

25. 원천등록기. S 다중선택기에 의하여 제공되는 원천연산수를 위하여 리용되는 등록기파일안의 주소

26. 원천등록기. R 다중선택기에 의하여 제공되는 원천연산수를 위하여 리용되는 등록기파일안의 주소

마지막으로 마당 27 은 8bit 조작코드로서 ALU 에서 수행되는 산수-론리연산기능을 규정한다. 표 15-9에는 가능한 조작을 목록으로 작성하여 제기하였다.

마당 17～27을 규정하는데 리용된 코드화의 실례로서 등록기1의 내용을 등록기 2에 더하여 그 결과를 등록기 3 에 넣는 명령을 보기로 하자. 아쎔블리언어명령은 다음과 같이 작성한다.

CONT11 [17], WELH, SELRFYMX, [24], R3, R2, R1, PASS+ADD

표 15-9. TI 8832 등록기화된 ALU 명령마당(마당 27)

그룹 1		기 능
ADD	H#01	R + S + C_n
SUBR	H#02	(NOT R) + S + C_n
SUBS	H#03	R = (NOT S) + C_n
INSC	H#04	S + C_n
INCNS	H#05	(NOT S) + C_n
INCR	H#06	R + C_n
INCNR	H#07	(NOT R) + C_n
XOR	H#09	R XOR S
AND	H#0A	R AND S
OR	H#0B	R OR S
NAND	H#0C	R NAND S
NOR	H#0D	R NOR S
ANDNR	H#0E	(NOT R) AND S
그룹 2		**기 능**
SRA	H#00	단정확도 산수오른쪽밀기
SRAD	H#10	배정확도 산수오른쪽밀기
SRL	H#20	단정확도 론리오른쪽밀기
SRLD	H#30	배정확도 론리오른쪽밀기
SLA	H#40	단정확도 산수왼쪽밀기
SLAD	H#50	배정확도 산수왼쪽밀기
SLC	H#60	단정확도 순환왼쪽밀기
SLCD	H#70	배정확도 순환왼쪽밀기
SRC	H#80	단정확도 순환오른쪽밀기
SRCD	H#90	배정확도 순환오른쪽밀기
MQSRA	H#A0	MQ 등록기 산수연산오른쪽밀기
MQSRL	H#B0	MQ 등록기 론리연산오른쪽밀기
MQSLL	H#C0	MQ 등록기 론리연산왼쪽밀기
MQSLC	H#D0	MQ 등록기 순환왼쪽밀기
LOADMQ	H#E0	MQ 등록기넣기
PASS	H#FO	ALU 를 지나 Y 에로(밀기조작 없이)
그룹 3		**기 능**
SET 1	H#08	비트 1 설정
SET 0	H#18	비트 0 설정
TB 1	H#28	비트 1 검사
TB 0	H#38	비트 0 검사
ABS	H#48	절대값
SMTC	H#58	부호크기/2 의 보수
ADDI	H#68	직접 더하기
SUBI	H#78	직접 덜기
BADD	H#88	S 에 R 바이트더하기

표계속

BSUBS	H#98	R 로부터 S 바이트덜기
BSUBR	H#A8	S 로부터 R 바이트덜기
BINCS	H#B8	S 의 바이트증가
BINCNS	H#C8	부정 S 의 바이트증가
BXOR	H#D8	R 와 S 의 바이트배타적론리더하기
BAND	H#E8	R 와 S 의 바이트론리곱하기
BOR	H#F8	R 와 S 의 바이트론리더하기
그룹 4		**기 능**
CRC	H#00	순환여유문자축적
SEL	H#10	S 혹은 R 의 선택
SNORM	H#20	단정확도길이정규화
DNORM	H#30	배정확도길이정규화
DIVRF	H#40	나누기의 나머지값길이고정
SDIVQF	H#50	부호 있는 나누기의 상길이고정
SMULI	H#60	부호 있는 곱하기반복
SMULT	H#70	부호 있는 곱하기완료
SDIVIN	H#80	부호 있는 나누기초기화
SDIVIS	H#90	부호 있는 나누기시작
SDIVI	H#A0	부호 있는 나누기반복
UDIVIS	H#B0	부호 없는 나누기시작
UDIVI	H#C0	부호 없는 나누기반복
UMULI	H#D0	부호 없는 곱하기반복
SDIVIT	H#E0	부호 있는 나누기완료
UDIVIT	H#FO	부호 없는 나누기완료
그룹 5		**기 능**
LOADFF	H#0F	나누기넣기/BCD 방아쇠
CLR	H#1F	지우기
DUMPFF	H#5F	나누기출구/BCD 방아쇠
BCDBIN	H#7F	BCD 부호를 2 진부호로 변환
EX3BC	H#8F	3 넘침부호(Excess_3)의 바이트교정
EX3C	H#9F	3 넘침부호(Excess_3)의 단어교정
SDIVO	H#AF	부호 있는 나누기의 자리넘침검사
BINEX3	H#DF	2 진부호를 3 넘침부호(Excess_3)로 변환
NOP32	H#FF	조작없음

기호번역프로그람은 이것을 2 진수로 된 해당한 기계어로 번역한다. 명령의 개별적요소들은 다음과 같이 서술할수 있다:

- CONT11 는 기본 NOP 명령이다.
- 마당 [17]은 32bit 등록기가 쓰기되도록 WELH(쓰기허락, L 과 H)로 된다.
- 마당 [18]은 SELRFYMX 로 변환되여 ALU Y MUX 출구로부터의 반결합을 선택한다.

- 마당 [24]는 목적등록기로서 등록기 R3 을 지적한다.
- 마당 [25]는 원천등록기의 하나로서 등록기 R2 을 지적한다.
- 마당 [26]은 원천등록기의 하나로서 등록기 R1 를 지적한다.
- 마당 [27]은 ALU 의 ADD 연산을 규정한다. ALU 의 밀기회로명령은 PASS 이며 따라서 ALU 출구는 밀기회로에 의하여 밀기되지 않는다.

기호표식에 대하여 몇가지 강조되여야 할것이 있다. 우선 련속마당에 대한 번호를 규정할 필요가 없다. 즉

CONT11 [17], WELH, [18], SELRFYMX

가 마당 18 안에 있기때문에

CONT11 [17], WELH, SELRFYMX

로 쓸수 있다.

다음으로 표 15-9 의 그룹 1 에 있는 ALU 명령들은 늘 그룹 2 와 함께 리용된다. 그룹 3-5 에 있는 ALU 명령은 그룹 2 와 함께 리용되지 말아야 한다.

제 5 절. 마이크로프로그람의 응용

미이크로프로그람이 도입된후 특히 1960 년대 후반기에 이르러서야 마이크로프로그람의 응용은 점차 광범하게 리용되게 되였다. 1971 년 초에는 마이크로프로그람의 리용이 뚜렷해 졌다. 현재 마이크로프로그람의 응용을 보면 다음과 같다.

콤퓨터의 실현

- 모의
- 조작체계지원
- 전용장치의 실현
- 고수준언어의 지원
- 마이크로고장진단
- 사용자재구성기술(마이크로프로그람을 필요에 따라 다시 짜넣는 기술)

이 장에서는 콤퓨터실현에 대하여 론의하였다. 마이크로프로그람방식은 조종장치실현을 위한 체계적인 기술을 제공한다. 그와 관련된 기술은 모의이다[MALL75]. 모의는

원래 다른 언어로 작성된 프로그람을 실행하기 위하여 하나의 기계에 마이크로프로그람을 리용하는것을 말한다. 모의의 가장 일반적인 리용은 한 콤퓨터에서 해야 할 일을 다른 콤퓨터에서 할수 있게 사용자를 돕는것이다. 이것은 흔히 많은 기계의것을 새 기계에서 실행할수 있게 함에 있어서 고객들의 편리를 도모할 목적밑에 제작회사에 의하여 실현된다. 한 연구사는 10년이상 물리적인 변화만 하여 온 IBM SYSTEM1370이 세상에 나온것은 1983년이였다고 주장하고 있다.[MALL83]

마이크로프로그람의 또 한가지 리용은 조작체계에 대한 지원이다. 마이크로프로그람은 조작체계쏘프트웨어의 중요한 부분들을 교체하는 기본단위를 실현하는데 리용할수 있다. 이 수법은 조작체계의 실현과 조작체계성능개선을 쉽게 할수 있다.

마이크로프로그람은 주콤퓨터와 결합할수 있는 **전용장치**를 실현할수 있는 수단으로서도 쓸모가 있다. 이에 대한 대표적인 실례가 자료통신기판이다. 이 기판에는 자체의 극소형처리장치가 장비되여 있다. 그것은 전용을 목적으로 하기때문에 성능을 개선하기 위한 쏘프트웨어를 리용하는것이 아니라 그 기능의 일부를 마이크로프로그람으로 실현하자는것이다.

마이크로프로그람의 단 하나의 응용분야는 고수준언어지원이다. 마이크로프로그람에서는 여러개 기능들과 자료형태들을 직접 실현할수 있다. 그 결과는 프로그람을 필요한 기계어형태로 번역하기가 쉽다는것이다. 결과적으로 기계어는 고수준언어(례하면 포트란, 코볼, 에이다)의 요구를 만족하게 해준다.

마이크로프로그람은 체계오유를 감시하고 검사하며 분리하고 수정하는것을 지원하는데 리용할수 있다. 이것을 **마이크로진단**이라고 하며 체계유지기능(오유없이 동작할수 있는 기능)을 강화할수 있다. 이 방법은 그것이 검출되였을 때 체계가 점차로 재구성할수 있게 한다. 례하면 만일 고속다중선택기가 비정상이면 마이크로프로그람 다중선택기가 대신 동작할수 있다.

마이크로프로그람응용의 또 하나의 일반적인 분류는 사용자재구성기술이다. 일부 기계들은 쓰기가능한 조종기억기(즉 ROM보다도 RAM으로 된 조종기억기)들을 가지고 있으므로 사용자들은 여기에 마이크로명령모임을 기억해 둔다. 이것은 사용자가 재구성하는데 요구되는 응용프로그람에 따라서 기계를 재구성한다.

마이크로프로그람의 또 하나의 응용은 **사용자구성기술**이다. 일부 기계들은 **쓰기가능한 조종기억기**(즉 ROM 보다도 RAM 으로 된 조종기억기)들을 가지고 있으므로 사용자들은 여기에 마이크로프로그람을 써넣을수 있다. 일반적으로 완전히 수평적이면서 리용하기 쉬운 마이크로명령들이 제공되여 있다.

이것을 리용하여 사용자는 요구되는 응용프로그람에 따라서 기계를 조종할수 있게 마이크로프로그람을 다시 짜넣을수 있다.

참고문헌

마이크로프로그람작성방법을 서술한 책들은 많다. 마이크로프로그람작성방법을 가장 폭 넓게 취급한것은 [LYNC93]이다. [SEGE91]은 단순한 16bit 처리장치의 단계별 설계방법에 의한 마이크로코드화된 체계의 설계와 마이크로코드작성의 근본원리들을 고찰하고 있다[CART96]. 역시 간단한 기계를 리용하는 기초적인 개념들을 취급하고 있다.

[PARK89]와 [TI90]은 TI8800 쏘프트웨어개발기판에 대하여 상세히 고찰하고 있다.

CART96 Carter, J. *Microprocessor Architecture and Microprogramming.* Upper Saddle River, NJ: Prentice Hall, 1996.

LYNC93 Lynch, M. *Microprogrammed State Machine Design.* Boca Raton, FL:CRC Press, 1993

PARK89 Parker, A., *and Hamblen, J. An Introduction to microprogramming with Exercises Designed for the Texas Instruments SN74ACT8800 Software Developmment Board.* Dallas, TX: Texas Instrum_ ents, 1989.

SEGE91 Segee, B., and Field, J. *Microprogramming and Computer Architecture.* New York: Wiley, 1991.

TI90 Texas Instruments Inc. *SN74ACT880 Family Data Manual.* SCSS006C, 1990

련습문제

1. 윌크스에 의하여 설계된 가상기계에서 곱하기명령의 실현을 해설과 흐름도를 리용하여 설명하시오.

2. 다음의 기호형태로 된 마이크로명령이 있다고 하자.

   IF (AC_0 = 1) THEN CAR ← ($C_{0\text{-}6}$) ELSE CAR ← (CAR) + 1

   여기서 AC_0 은 축적등록기의 부호비트이며 $C_{0\text{-}6}$ 은 마이크로명령의 첫 6 개의 비트이다. 이 마이크로명령을 리용하여 AC 가 부인 경우 갈라 지는(분기되는) 분기등록기덜기(BRM)기계명령을 실현하는 마이크로프로그람을 작성하시오. 마이크로명령의 비트 C_1-C_n 은 마이크로조작에서 병렬로 조작되는 비트들이라고 본다.

프로그람을 기호로 표시하시오.

3. 하나의 처리장치가 자기의 명령주기에 4 개의 기본주기 즉 꺼내기, 간접, 실행 및 새치기를 가지고 있다. 하드웨어를 실현함에 있어서 두개의 1bit 기발이 현재 집행되는 주기를 지적한다.

ㄱ. 왜 이 기발이 필요한가?

ㄴ. 왜 이 기발들이 마이크로조종장치안에는 필요 없는가?

4. 그림 15-7 의 조종장치를 고찰해 보자. 조종기억기가 24bit 폭이고 마이크로명령형식의 조종부분이 두개 마당으로 나뉘여 져 있으며 13bit 가운데서 하나의 마이크로조작마당으로써 수행되여야 할 마이크로조작들을 규정한다.

주소선택마당은 기발에 기초한 조건을 규정하는데 이 조건은 마이크로명령의 분기를 일으키자는데 있다. 기발은 8 개이다.

ㄱ. 주소선택마당안에 얼마만한 비트가 있어야 하는가(주소선택마당이 몇 비트로 되여야 하는가)

ㄴ. 주소마당은 몇비트로 되여야 하는가?

ㄷ. 조종기억기의 용량은 얼마인가?

5. 앞의 문제의 조건에서 무조건분기는 어떻게 수행되는가? 분기를 어떻게 피할수 있는가(즉 분기, 조건 혹은 무조건을 규정하지 않는 마이크로명령을 서술하시오)?

6. 매개 기계명령루틴을 위하여 8 개의 조종단어가 제공된다고 하자. 기계명령조작코드가 5bit 이고 조종기억기가 1024 단어를 가진다면 명령등록기로부터 조종주소등록기에로의 배치를 설명하시오.

7. 부호화된 마이크로명령형식이 리용된다고 하자. 9bit 의 마이크로조작마당이 46 개의 서로 다른 동작을 규정하는 보조(부분)마당으로 나뉘여 질수 있다는것을 설명하시오.

8. 16 개의 등록기, 16 개의 론리 및 16 개의 산수연산기술을 가진 ALU, 8 개의 조작을 할수 있는 밀기등록기, 이들모두를 내부처리장치모선에 련결시킨 처리장치가 있다고 하자. 처리장치가 할수 있는 여러가지 마이크로조작을 규정하는 마이크로명령형식을 설계하시오.

제 5 편. 병렬조직

제 5 편의 중심

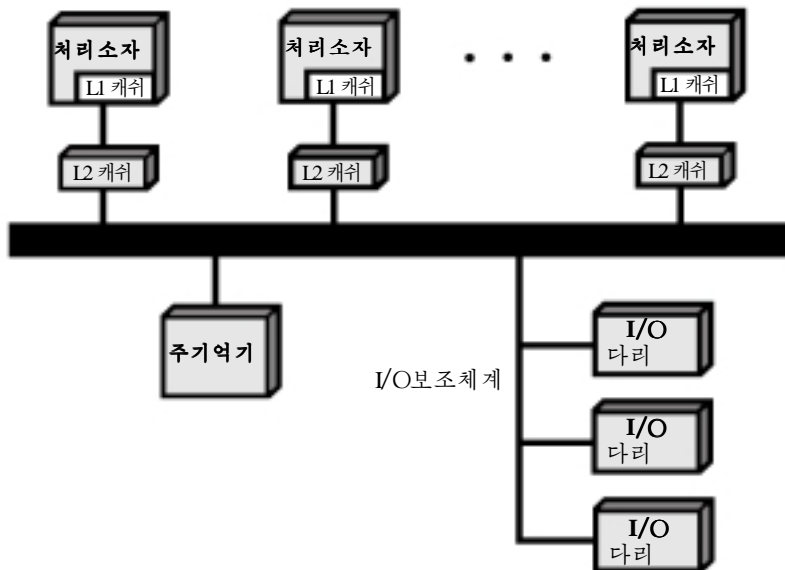
제 5 편에서는 병렬처리조직의 주요부분에 대하여 고찰한다. 병렬처리조직에서 다중처리장치들은 서로 협동하여 응용프로그람을 실행한다. 슈퍼스칼라처리장치는 명령준위에서 병렬실행을 할수 있기때문에 병렬처리조직에서는 다중처리장치에 의하여 작업이 병렬로 그리고 협동적으로 진행되도록 하기 위한 여러준위병렬화를 고찰하게 된다. 이러한 내부조직으로 하여 여러가지 문제가 제기된다. 례하면 개개가 자체의 캐쉬를 가지고 있는 다중처리장치들이 같은 기억기에 대한 접근을 공유하자면 하드웨어 혹은 쏘프트웨어기구는 두 처리장치가 주기억기의 유효영상을 공유해야 하는데 이것을 캐쉬일치성문제라고 한다. 이 설계항목과 기타 문제를 제 5 편에서 취급한다.

제 5 편의 장별 내용

제 16 장. 병렬처리

제 16 장에서는 병렬처리를 개괄하였다. 여기서는 다중처리장치콤퓨터를 구성하기 위한 세가지 방안인 대칭다중처리장치콤퓨터, 클라스터와 비균일기억기접근(NUMA)콤퓨터들을 고찰한다. 다중처리장치를 구성하여 성능을 높이기 위한 두가지 가장 대표적인 방법은 대칭처리장치와 클라스터를 구성하는것이다. 비균일기억기접근체계는 아직 광범하게 상업적성과를 달성하지 못하였지만 최근에 가장 중시되는 개념이다. 마지막으로 제 16 장에서는 벡토르처리장치의 특수한 구성에 대하여 고찰하게 된다.

제 16 장. 병렬처리

◆◆◆

◆ 체계의 성능을 높이기 위한 전통적인 방법은 주어 진 작업부하를 지원하기 위하여 병렬로 실행할수 있는 다중처리장치를 리용하는것이다. 그가운데서 가장 대표적인 두가지 다중처리장치조직은 대칭다중처리장치조직과 클라스터조직이다. 최근에 비균일기억기접근체계가 많이 리용되고 있다.

◆ 대칭다중처리장치체계는 같은 콤퓨터안에서 여러개의 같은 처리장치들이 모선이나 몇가지 종류의 교환체계에 의하여 호상 련결된 구성을 말한다. 대칭다중처리장치의 주소화에서 제일 심중한 문제는 캐쉬의 일치성문제이다. 매개 처리장치는 자체의 캐쉬를 가지고 있으므로 주어 진 행의 자료가 여러개의 캐쉬안에 존재할수 있다. 만일 그러한 행이 한 캐쉬안에서 변화되면 그때 두개의 주기억기와 다른 캐쉬는 그 행에 대하여 무효인 형식을 취한다. 캐쉬일치성규약은 이 문제를 극복하기 위하여 설계된것이다.

◆ 다중콤퓨터체계는 하나의 자원을 함께 처리할수 있게 여러대의 콤퓨터를 호상 련결하여 하나로 묶어 놓은것으로서 마치도 한대의 콤퓨터가 처리하는 것과 같은 느낌이 든다.

◆ 비균일기억기접근체계는 주어 진 처리장치로부터 기억기안의 단어까지 접근시간이 기억단어의 주소에 따라서 변하는 공유기억기식다중처리장치를 말한다.

◆◆◆

전통적으로 콤퓨터는 순차기계로 고찰되여 왔다. 대부분의 프로그람작성언어는 프로그람작성자가 명령의 순차에 따라서 알고리듬을 규정할것을 요구한다. 처리장치는 한번에 하나씩 차례로 기계어명령을 실행하는 방법으로 프로그람을 집행한다. 매개 명령은 명령꺼내기, 연산수꺼내기, 연산수행, 결과기억의 순서로 실행된다.

콤퓨터에 대한 이러한 고찰은 완전히 정확한것이라고 볼수 없다. 마이크로조작준위에서 다중조종신호들은 같은 시간에 발생된다. 명령관흐름은 최소한 명령꺼내기와 연산실행을 동시에 하는것으로 오래전부터 잘 알려 져 있다. 이것들도 둘 다 병렬로 기능을 수행하는 방식들이다. 이 방식은 후에 명령준위병렬화를 리용한 슈퍼스칼라방식으로 발전하였다. 슈퍼스칼라기계는 하나의 처리장치안에 여러개의 실행장치를 가지고 있으며 따라서 같은 프로그람안에 있는 여러개의 명령을 동시에 실행할수 있다.

콤퓨터기술이 발전하고 콤퓨터하드웨어의 값이 내려감에 따라 콤퓨터설계자들도 성능을 높이기 위하여 일부 경우에 리용가치를 높이도록 더욱더 병렬화를 모색하고 있다. 이 장에서는 총적고찰을 먼저 진행하고 그다음 병렬화조직의 가장 우수한 방식가운데서 세가지를 고찰한다. 우선 초시기부터 리용되여 현재까지도 가장 대표적인 병렬화의 실례로 되는 대칭다중처리장치체계를 본다. 대칭다중처리장치조직에는 공통기억기를 공유하는 다중처리장치들을 리용한다. 여기서는 캐쉬의 일치성문제가 제기된다. 다음으로 개별콤퓨터들을 여러개 협동방식으로 동작하게 구성한 클라스터체계를 서술한다. 클라스터체계는 하나의 대칭콤퓨터체계로서 담당할수 없는 작업부하를 지원하는 일반적인 체계로 점차 리용되고 있다. 마지막으로 고찰하게 되는 다중처리장치리용방식은 비균일기억기접근기계이다. 비균일기억기접근기계들은 아직 상대적으로 생소하며 널리 알려 지지 않았지만 대칭처리장치방식과 병렬콤퓨터방식의 우점을 받아 들인것

으로 고찰하고 있다. 이 장에서는 또한 벡토르콤퓨터에 대한 하드웨어조직방법에 대하여 보게 된다. 이 방식들은 벡토르 혹은 류점수의 배렬을 처리하기 위하여 ALU 를 최적화한다. 이것들은 슈퍼콤퓨터체계를 실현하는데 리용된다.

제 1 절. 다중처리장치조직

1. 병렬처리장치체계의 류형

Flynu[FLYN72]에 의하여 처음으로 도입된 분류방법은 병렬처리능력을 가진 체계를 분류하는 가장 일반적인 방법이다. Flynu 은 다음과 같이 콤퓨터체계를 분류할것을 제기하였다.

- **단일명령,단일자료(SISD)흐름:** 하나의 처리장치가 하나의 기억기안에 기억된 자료에 대한 조작인 하나의 명령흐름을 실행한다. 단일처리장치들은 이 부류에 속한다.
- **단일명령,다중자료(SIMD)흐름:** 하나의 기계명령이 여러개의 처리요소들의 실행을 동시에 조종한다. 매개 처리요소들은 련상자료기억기를 가지고 있으며 그리하여 매개 명령이 서로 다른 처리장치들에 의하여 각이한 모임자료로 실행된다. 벡토르처리장치와 배렬처리장치들이 이 부류에 속한다.

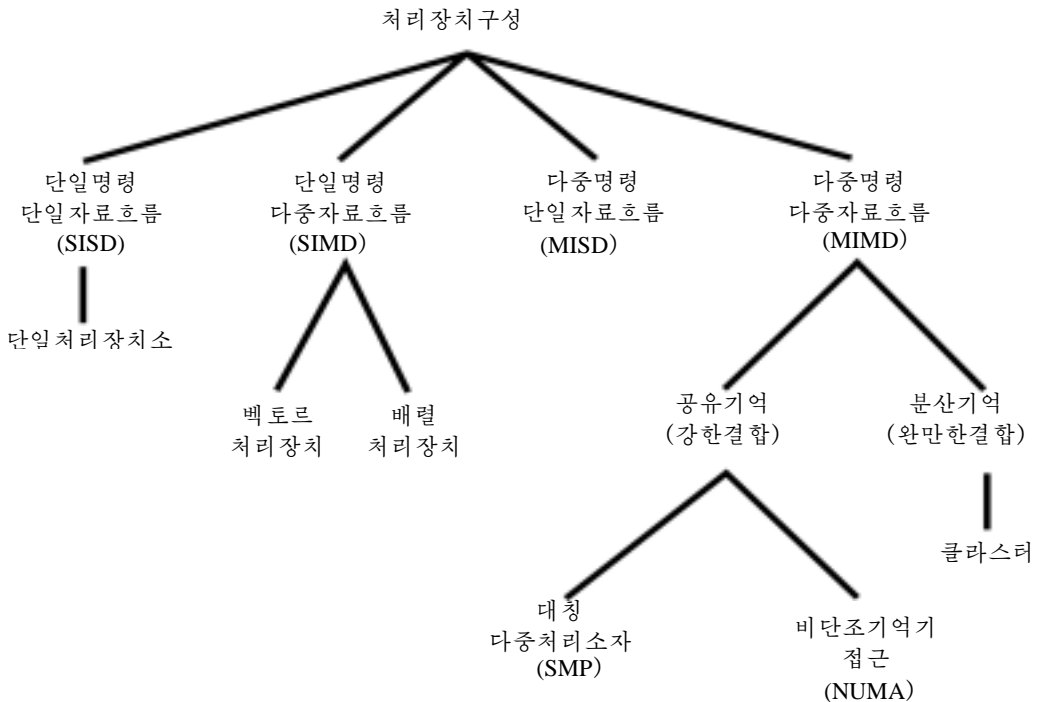



그림 16-1. 병렬처리장치구성방식의 분류

- **다중명령, 단일자료(MISD)흐름:** 자료가 차례로 여러개의 처리장치에로 전송되며 개개의 처리장치는 서로 다른 명령들을 실행한다. 이 구조는 아직 실현되지 못하였다.
- **다중명령, 다중자료(MIMD)흐름:** 여러개의 처리장치들이 자료에 따라 서로 다른 명령을 실행한다. 대칭처리장치, 클라스터 그리고 비균일기억기접근체계들이 여기에 속한다.

MIMD 조직에서 처리장치들은 일반목적으로 리용되며 이 개개의 처리장치들은 해당한 자료변환을 수행하는데 필요한 모든 명령을 수행할수 있다. MIMD는 처리장치가 통신하는 방법에 따라 더 세분화된다(그림 16-1). 만일 처리장치들이 공통기억기를 공유한다면 그때 매개 처리장치들은 공유기억기안에 기억된 프로그람과 자료를 접근할수 있으며 그 기억기를 통하여 서로 통신을 한다. 이러한 체계가운데서 제일 일반적인 형태는 **대칭다중처리장치**체계이며 이에 대하여서는 제 2 절에서 론의한다. 대칭다중처리장치체계에서는 여러개의 처리장치들이 공유모선 혹은 다른 호상련결기구에 의하여 하나의 기억기나 완충기억기를 공유한다. 구별되는 특징은 기억기의 임의의 구역에 대한 기억접근시간이 매개 처리장치에서 거의 같다는것이다. 최근에 개발된 체계는 **비균일기억기접근**으로서 제 5 절에서 서술하였다. 이 구성은 이름이 보여 주는것처럼 기억기의 서로 다른 구역에 대한 기억접근시간이 비균일기억기접근처리장치에 따라서 다르다는것이다.

독립적인 단일처리장치 혹은 대칭다중처리장치들을 묶으면 클라스터가 될수 있다. 콤퓨터내에서의 통신은 고정된 경로나 망을 통하여 실현된다.

2. 병렬처리조직

그림 16-2는 그림 16-1에서 보여 준 분류에 따르는 일반구성들을 보여 준것이다. 그림 16-2 ㄱ는 SISD의 구조를 보여 준것으로서 여기에는 명령흐름을 처리장치에 제공하는 몇개의 조종장치가 있다. 처리장치들은 기억장치로부터 나오는 하나의 자료흐름을 조작한다. SIMD에는 단일한 명령흐름을 여러가지 처리요소들에 공급하는 조종장치가 하나 있다. 매개 처리요소들은 자기자체를 위한 기억기를 가지고 있거나(그림 16-2 ㄴ) 혹은 공유기억기를 가지고 있다. 마지막으로 MIMD에서는 서로 다른 명령흐름을 그자체의 처리요소에 각각 공급하는 여러개의 조종장치가 있다. MIMD는 공유기억기를 가진 다중처리장치체계나 분산기억기를 가진 다중콤퓨터체계이다(그림 16-2 ㄹ).

대칭다중처리장치체계, 클라스터체계, 비균일기억기접근체계에 대한 설계인 경우 물리적구성, 호상련결구조, 처리장치들사이의 통신, 조작체계설계, 응용프로그람기술과 관련된 문제들을 비롯하여 복잡한 항목들이 있다. 여기에서는 조작체계설계와 관련된 항목을 간단히 소개하고 주로 구성에 주목을 돌리려고 한다.

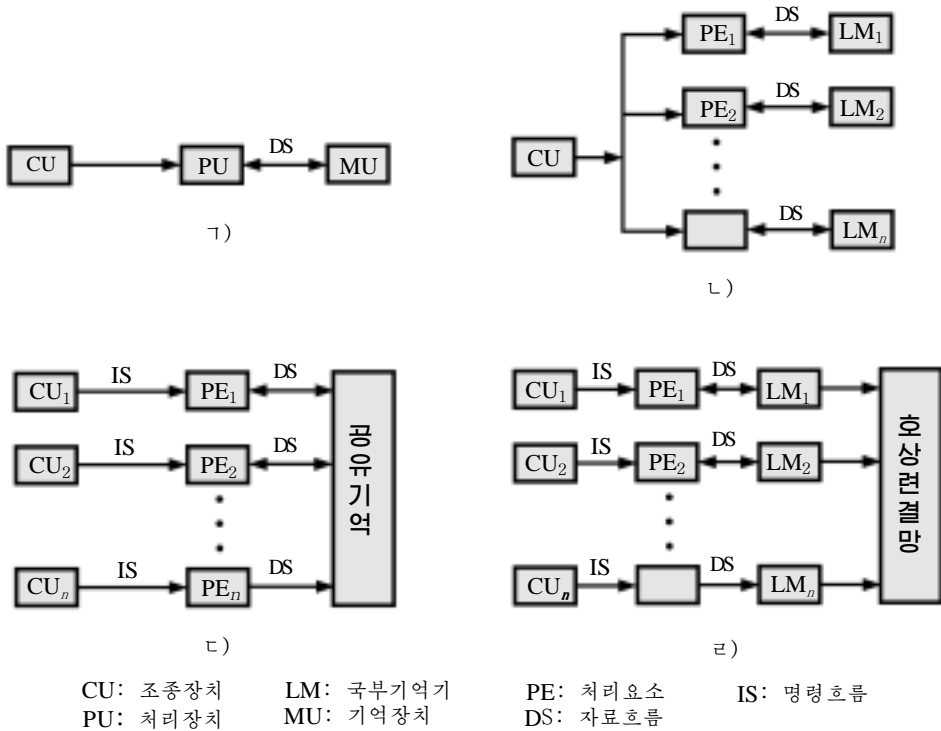



그림 16-2. 선택적인 콤퓨터구조
ㄱ-SISD, ㄴ-SIMD(분산기억기를 가지는 경우), ㄷ-MIMD(공유기억기를 가진), ㄹ-MIMD(분산기억기를 가진)

제 2 절. 대칭다중처리장치

최근까지만 하여도 모든 단일사용자개인용콤퓨터들과 대부분의 워크스테이션(WS)에 한개의 일반용극소형처리장치를 리용하였다. 성능에 대한 요구가 높아 지고 극소형처리장치의 가격이 계속 낮아 지는데 따라 제작회사들은 대칭다중처리장치체계들을 도입하기 시작하였다. 대칭다중처리장치체계는 콤퓨터하드웨어구성방식과 그 방식에 영향을 주는 조작체계의 동작에 많이 관계된다. 대칭다중처리장치체계는 다음과 같은 특성을 가진 콤퓨터체계로 정의된다.

- 둘이상의 처리장치들이 있다.
- 이 처리장치들은 같은 주기억기와 I/O 장치를 공유하며 모선 혹은 다른 내부련결기구에 의하여 호상 련결된다. 기억기접근시간은 매개 처리장치들에서 거의 같다.
- 모든 처리장치들은 같은 통로 아니면 같은 장치에 경로를 제공하는 다른 통로를 통하여 I/O 장치들에 대한 접근을 공유한다.
- 모든 처리장치들은 같은 기능(여기서 **대칭성**이라는 의미)을 수행한다.
- 체계는 일감, 과제, 파일, 자료요소의 준위에서 처리장치와 프로그람사이 호상작용을 제공하는 통합조작체계에 의하여 조종된다.

1~4까지는 전체 해설이다. 5는 클라스터와 같이 결합이 강하지 않은 극소형처리장치와의 대조관계를 보여 준것이다. 후자인 경우에 호상작용을 위한 물리적단위는 일반적으로 통보문이나 옹근 하나의 파일이다. 대칭다중처리장치체계에서 개별적자료요소들은 호상작용을 위한 준위를 구성할수 있으며 처리장치들사이에서는 높은 급의 호상협동이 진행된다.

대칭다중처리장치의 조작체계는 모든 처리장치들에 대하여 처리공정이나 스레드(Thread)들의 순서를 작성한다. 대칭다중처리장치체계는 처리장치가 하나인 극소형처리장치체계에 비하여 다음과 같은 우점이 있다.

- **성능:** 콤퓨터에 의하여 진행되는 작업을 몇개 부분이 병렬로 진행될수 있도록 구성할수 있다면 그때 다중처리장치로 된 체계는 같은 류형의 하나의 처리장치로 된 체계보다 더 큰 성능을 얻을수 있다(그림 16-3).
- **리용률:** 대칭다중처리장치체계에서 모든 처리장치들은 같은 기능을 수행하기때문에 어느 한 처리장치가 고장이 나도 콤퓨터체계는 정지되지 않는다. 그대신 체계는 성능이 떨어 지지만 동작은 계속할수 있다.
- **성능제고:** 사용자는 처리장치를 보충하는 방법으로 체계의 성능을 높일수 있다.
- **생산규모:** 제작자들은 체계안에 구성된 처리장치의 수에 따라서 가격과 성능에서 서로 다른 제품계렬을 만들수 있다.

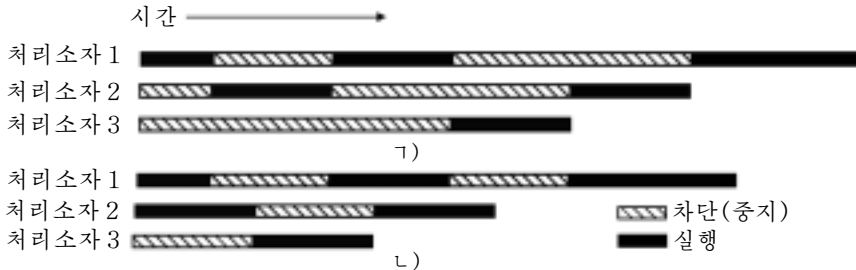



그림 16-3. 다중처리소자
ㄱ-간격식, ㄴ-간격식과 겹칩식

이것들은 담보에 의해서라기보다도 잠재적인 리익을 준다는것을 고려하는것이 중요하다. 조작체계는 대칭다중처리장치체계에서 병렬화를 실현하는데 필요한 도구들과 기능들을 제공해 주어야 한다.

대칭다중처리장치체계의 매력적인 특징은 다중처리장치들의 존재가 사용자에게 명백하다는것이다. 조작체계는 스레드나 처리공정에 대한 처리순서작성과 처리장치들사이 동기화에 주의를 돌려야 한다.

1. 조직

그림 16-4에는 다중처리장치체계의 일반적구성을 보여 주었다. 여기에는 두개 혹은 그이상의 처리장치들이 있다. 매개 처리장치들은 조종장치, ALU, 등록기들과 캐쉬들로

구성되여 있다. 매개 처리장치들은 여러가지 형태의 호상결합기구를 통하여 공유주기억기와 I/O 장치를 접근한다. 처리장치들은 주기억기를 통하여 공동자료구역안에 남아 있는 통보문과 상태정보들을 서로 통신할수 있다. 또한 처리장치들이 신호를 직접 교환할수도 있다. 기억기는 흔히 여러개의 기억블로크에 대한 접근이 다중으로 그리고 동시에 가능하도록 구성한다. 일부 구성에서는 매개 처리장치가 공유되여 자체의 특권주기억과 I/O 통보들을 가질수 있게 되여 있다.

다중처리장치의 조직은 다음과 같이 분류할수 있다.

- 시분할 및 공통모선
- 다중포구기억기
- 중앙조종장치

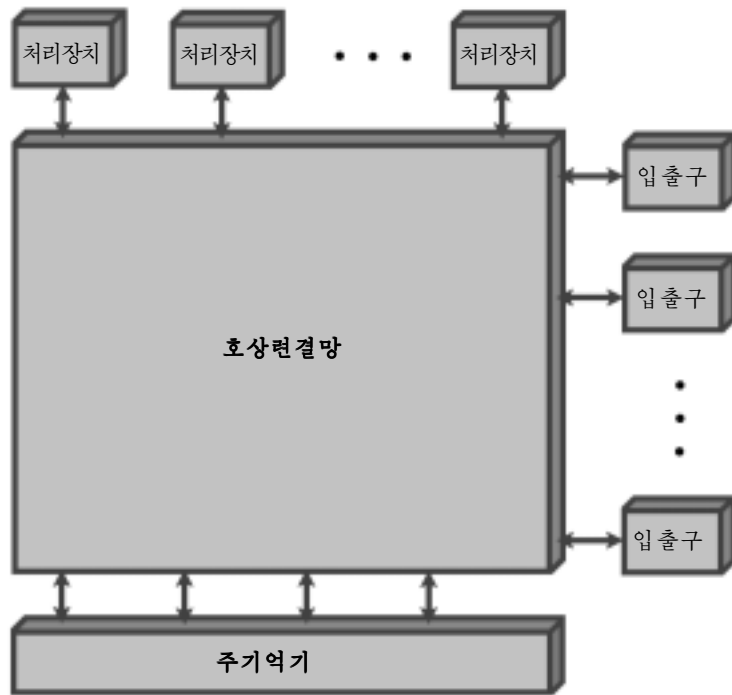



그림 16-4. 여러개의 처리장치를 결합한 다중처리장치의 일반적구성도

시분할모선

시분할모선은 다중처리체계를 구성하는데서 제일 간단한 기구이다(그림 16-5). 구조와 대면부는 기본적으로 모선호상결합을 리용하는 단일처리장치체계와 거의 같다. 모선은 조종선, 주소선 그리고 자료선들로 되여 있다. I/O처리장치들로부터의 직접기억기접근(DMA)전송을 실현하기 위하여 다음과 같은 특징을 갖추어야 한다.

- **주소지정:** 자료의 원천주소와 목적주소를 결정하기 위하여 모선에 련결된 모듈들을 구별할수 있어야 한다.

- **중재:** 임의의 I/O 모듈은 림시로 《주》로서의 기능을 수행할수 있다. 몇개 종류의 우선권방식을 리용하여 모선조종에 대한 경쟁요구를 중재할수 있다.
- **시분할:** 하나의 모듈이 모선을 조종할 때 다른 모선은 조종되지 말아야 하며 관건되여 있어야 한다. 필요하다면 모선접근이 실현될 때까지 조작을 중단한다.

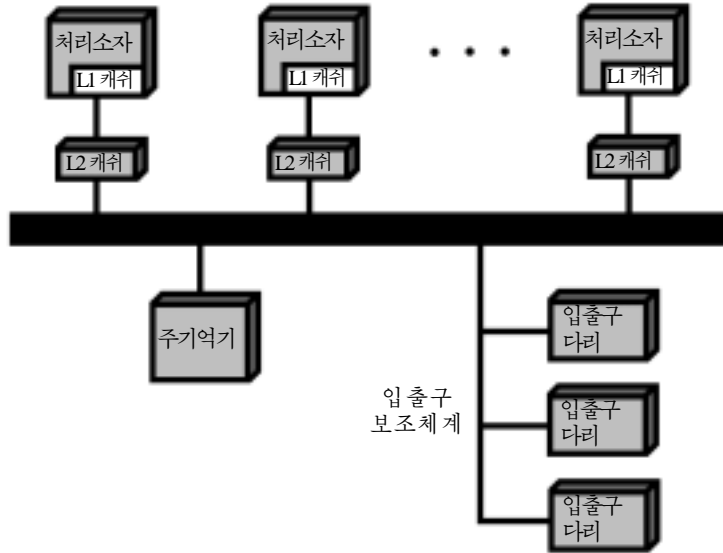



그림 16-5. 대칭다중처리소자구성

단일처리장치의 특징들을 직접 대칭다중처리장치체계에 리용할수 있다. 뒤에서 보게 되겠지만 모선을 통하여 하나 혹은 그이상의 기억모듈을 접근하는 다중 I/O 처리장치체계와 같은 다중처리장치체계들도 있다.

모선조직은 다른 조직에 비하여 다음과 같은 우점을 가진다.

- **간결성:** 다중처리장치조직가운데서 제일 간단한 방식이다. 매개 처리장치의 물리적대면부와 주소지정, 중재, 시분할론리는 하나의 처리장치체계와 같다.
- **유연성:** 모선에 여러개의 처리장치들을 더 련결하는 방법으로 체계를 확장하기가 일반적으로 쉽다.
- **믿음성:** 모선은 주로 피동매체이므로 여기에 련결된 임의의 장치에서의 오유는 체계전반에 영향을 주지 않는다.

모선조직에서의 기본결함은 성능이다. 모든 기억참조는 공통모선을 통하여 진행된다. 따라서 체계의 속도는 모선주기시간의 제한을 받는다. 성능을 개선하기 위하여 매개 처리장치는 캐쉬를 가져야 한다. 이것은 모선접근회수를 극히 줄일수 있다. 전형적으로 WS와 개인용콤퓨터의 대칭다중처리장치들은 두개 준위로 된 캐쉬를 가지고 있는데 하나는 L1(1준위)캐쉬로서 내부용이며 다른 하나는 L2(2준위)캐쉬로서 내부 혹은 외부용이다.

캐쉬를 리용함으로써 설계에서는 몇가지 새로운것을 고찰하게 된다. 매개 국부캐쉬들은 기억기의 일부분을 영상으로 가지고 있기때문에 만일 단어가 한 캐쉬안에서 변화되면 다른 캐쉬안에 있는 자료가 무시될수 있다. 이것을 막기 위하여 다른 처리장치들은

자료의 량갱신이 일어 나는것을 감시하여야 한다. 이것을 **캐쉬의 일치성문제**라고 하며 일반적으로 조작체계에 의해서가 아니라 하드웨어에 의하여 진행된다. 이 문제에 대하여서는 제 3절에서 취급한다.

다중포구기억기

다중포구기억기방식은 주기억기에 대한 접근을 매개 처리장치와 입출력모듈에 의하여 직접 그리고 독립적으로 할수 있는 방식이다(그림 16-6). 기억기와 관련된 론리에서는 충돌문제를 풀기 위한것이 반영되여 있다. 충돌을 막기 위하여 흔히 리용되는 방법은 매개 기억포구에 항구적으로 규정된 우선권을 할당하는것이다. 특히 매개 포구에서 물리적 및 전기적대면부는 마치도 하나의 포구들로 된 기억모듈과 같다. 따라서 처리장치나 I/O 모듈을 다중포구기억기에 적응시키는데는 거의 변경이 필요 없다.

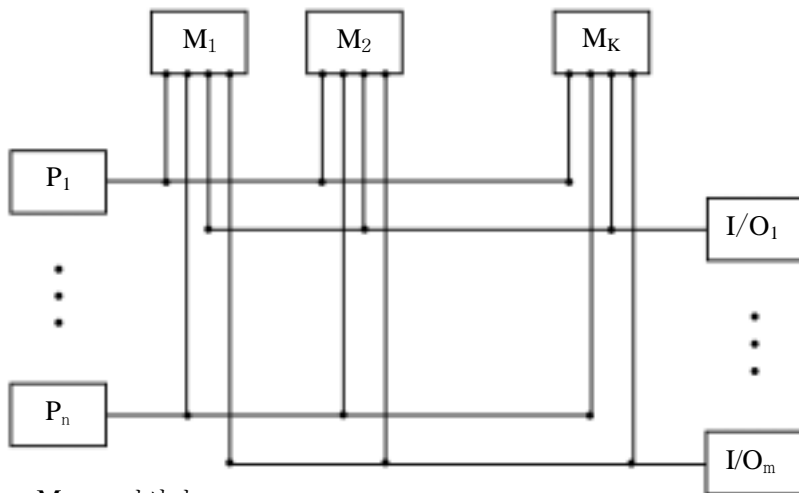



그림 16-6. 다중포구기억기

다중포구기억기방식은 기억체계에 필요한 론리가 매우 적은 모선방식에 비하여 더 복잡하다. 그러나 성능은 더 높은데 그 리유는 매개 처리장치가 개개의 기억모듈에 대응하는 통보를 가지고 있기때문이다. 다중포구의 다른 하나의 우점은 기억기의 일부분을 하나 혹은 여러개의 처리장치든가 I/O 모듈에서 전용으로 쓸수 있게 구성할수 있는것이다. 이것은 비법접근에 대한 안전성을 높일수 있으며 다른 처리장치에 의하여 변경할수 없는 기억구역안의 회복루틴들을 기억시킬수 있다.

또 다른 한가지 측면은 동시쓰기수법을 캐쉬조종에 리용할수 있는데 그 리유는 다른 처리장치가 기억갱신을 감시하는 경우 이 보다 더 편리한 수단이 없기때문이다.

중앙조종장치

중앙조종장치는 개별적인 모듈 즉 처리장치, 기억기, I/O 들사이에 개별적인 자료흐름의 입출력을 조종한다. 조종장치는 요구신호들을 완충하며 중재와 동기기능을 수행한다. 그리고 처리장치들사이에 상태정보나 조종정보를 통과시키며 캐쉬의 갱신을 감시한다.

다중처리장치의 조직을 조종하는 모든 론리들은 중앙조종장치안에 집중되여 있기때문에 I/O, 기억기와 처리장치와의 대면부는 그대로 남아 있다. 이것이 모선방식의 대면부에 유연성과 간결성을 준다. 이 방식의 기본결함은 조종장치가 매우 복잡한것인데 이것은 성능을 높이는데서 잠재적인 난점으로 되고 있다.

이 중앙조종장치의 구조는 IBM S/370 계렬과 같은 다중처리장치로 된 대형콤퓨터에 리용되였다. 이러한것은 오늘날 드물게 볼수 있다.

2. 다중처리장치를 위한 조작체계설계에 대한 고찰

대칭다중처리장치(SMP)를 위한 조작체계는 사용자가 체계자원을 조종하는 단일조작체계를 쓰는것처럼 인식하도록 처리장치와 다른 콤퓨터의 자원을 관리한다. 사실상 이러한 구성은 마치도 단일처리장치-다중프로그람체계처럼 인식된다. 대칭병렬처리장치와 단일처리장치의 두 경우에 다중일감이나 프로쎄스들은 한번에 처리될수 있으며 그것들의 실행순서를 규정하고 자원을 할당하는것은 조작체계가 할 일이다. 사용자는 하나의 처리장치가 하든 여러개의 처리장치가 하든간에 관계없이 다중프로쎄스나 프로쎄스안의 다중스레드들을 리용하는 응용프로그람을 작성할수 있다. 따라서 다중처리장치조작체계는 다중프로그람체계의 모든 기능과 다중처리장치에 적합한 특징들을 갖추고 있어야 한다. 기본설계항목은 다음과 같다.

- **동시병행처리:** OS 루틴들은 여러개 처리장치들이 같은 IS 코드들을 동시에 실행하게 하는것이다. OS 의 같은 부분과 서로 다른 부분을 실행하는 다중처리장치들을 리용하여 중지상태라든가 무효조작을 피하기 위하여 OS의 표들과 관리구조들을 일정하게 관리하여야 한다.
- **처리순서작성:** 처리순서작성은 처리장치에 의하여 수행된다. 그래야 충돌을 피할수 있다. 순서작성기는 준비된 프로쎄스장치들을 필요한 처리장치에 제공한다.
- **동기화:** 공유주소공간이든가 공유 I/O 자원에 대한 잠재적인 접근을 하는 다중능동처리장치에 있어서 효과적인 동기화에 주의를 돌려야 한다. 동기화는 호상독점적으로 사건이 처리되게 순서를 결정하는 기구이다.
- **기억관리:** 다중처리장치체계에서 기억관리는 제 7 장에서 론의된것처럼 단일처리장치기계에서 고찰한 모든 항목들을 취급하여야 한다. 그외에 조작체계는 최대의 성능을 얻기 위하여 다중포구기억기와 같은 필요한 하드웨어병렬화를 리용하해야 한다. 서로 다른 처리장치에 대한 페지화기구는 여러개의 처리장치들이 페지와 토막을 공유할 때 일치성의 실현과 페지교체를 조종한다.
- **믿음성과 오유조종:** 조작체계는 처리장치오유에 따르는 성능저하를 막아야 한다. 순서작성기와 조작체계의 다른 부분들은 처리장치의 오유를 인식하고 그에 따라 관리표들을 재구성하여야 한다.

3. 대칭다중처리장치로 된 대형콤퓨터(SMP)

대부분의 개인용콤퓨터들과 대칭다중처리장치로 된 워크스테이션(WS)은 그림 16-5에서 제시한것처럼 모선호상결합방식을 리용하고 있다. 여기서는 최근에 개발된 IBM

S/390 대형콤퓨터계렬에 리용되고 있는 한가지 방안에 대하여 보기로 하자[MAK97]. 그림 16-7 은 S/390 대칭다중처리장치방식의 전반적구성을 보여 주었다. 이 계렬의 체계들은 하나의 주기억기판으로 된 단일처리장치로부터 10 개의 처리장치와 4개의 기억기판으로 구성된 고성능체계까지 가지고 있다. 여기에는 I/O 처리장치로서의 기능을 수행하는 처리장치가 한두개 더 보충된다. 이 체계구성의 기본요소들은 다음과 같다.

- **처리장치(PU):** 이것은 CISC 극소형처리장치로서 가장 많이 쓰이는 명령은 장치배선론리식으로 되여 있으며 나머지는 마이크로프로그람방식으로 실행된다. 매개 처리장치에는 64KB 의 1 준위캐쉬가 내장되여 있으므로 접근할수 있다.
- **L2(2 준위)캐쉬:** 매개 2 준위캐쉬는 용량이 384KB 이다. 2 준위캐쉬들은 두개씩 짝을 무어 배치되여 있으며 매개 짝은 세개의 처리장치를 지원하며 한편으로 전체주기억공간을 접근할수 있다.
- **모선교환망다리(BSN):** 이 다리는 L2 캐쉬와 주기억을 련결시킨다. 매개 모선교환망다리는 2MB 크기의 3 준위(L3)캐쉬를 내장하고 있다.
- **기억기판:** 매개 기판의 기억용량은 8GB 이며 전체 용량은 32GB 이다.

S/390 대칭다중처리장치구성에는 흥미 있는 몇가지 특징이 있다. 그에 대하여 차례로 보기로 하자.

- 전환호상교환결합
- 2 준위(L2)공유캐쉬
- 3 준위(L3)캐쉬

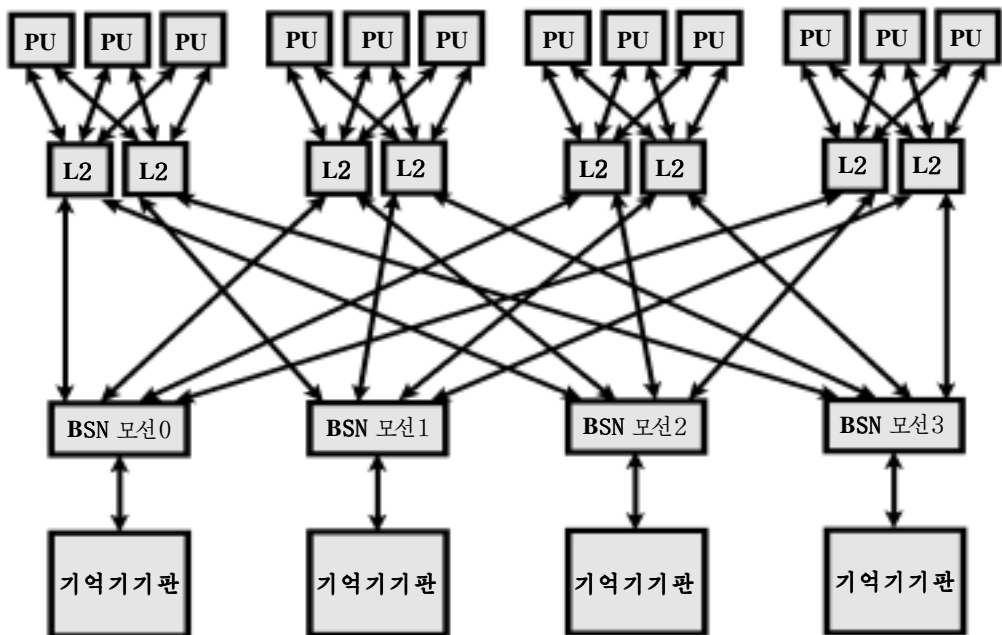



그림 16-7. IBM S/390 SMP 구성

절환호상결합

PC 와 WS 를 위한 대칭다중처리장치체계에서는 하나의 공유모선을 공동으로 리용한다(그림 16-5). 단일모선을 이렇게 배렬하면 설계의 스칼라성에 영향을 주는 어려운 문제가 제기된다. S/390 에서는 이 문제를 두가지 방법으로 해결하였다. 첫째로, 주기억을 4 개의 기판으로 나누고 그 개개에 기억기접근을 고속으로 조종할수 있는 자체의 기억조종장치를 넣었다. 이렇게 하면 주기억에 대한 평균전송부하는 1/4 로 줄어 드는데 그 리유는 기억기를 4개의 부분으로 나누고 4개의 독립적인 통로에 련결시켰기때문이다. 둘째로, 처리장치들로부터(실제로는 L2 캐쉬로부터) 하나의 기억기판까지의 련결은 공유모선의 형태로가 아니라 오히려 지점 대 지점련결이며 개개의 통로 L2 캐쉬를 통하여 세개의 처리장치로 된 그룹을 모선교환망다리에 련결한다. 모선교환망다리는 차례로 5개의 통로(4 개의 L2 통로, 한개의 기억기판)안의 자료를 절환할수 있는 스위치의 기능을 수행한다.

4 개의 L2 통로에 대하여 모선교환망다리는 4 개의 물리적통로를 하나의 론리자료모선에 련결한다. 따라서 4개의 L2 통로가운데서 어느 한 통로로 들어 오는 신호는 나머지 세개의 L2 통로로 되돌아 오게 된다. 이 특징은 캐쉬의 일치성을 지원한다.

주의해야 할것은 비록 4 개의 개별적인 기억기판이 있다고 하여도 매개 처리장치와 L2 캐쉬는 주기억쪽으로 두개의 물리적포구들만 가진다는것이다. 그 리유는 매개 L2 캐쉬는 오직 주기억기의 절반으로부터 자료를 꺼내여 보관하기때문이다. 주기억전체를 봉사하는데 두개의 캐쉬를 쓰며 매개 처리장치는 쌍으로 두개의 캐쉬에 련결하여야 한다.

공유 L2 캐쉬(2 준위공유캐쉬)

대칭다중처리장치체계에 쓰이는 전형적인 2 준위캐쉬방식에서 매개 처리장치는 전용 L1, L2 캐쉬를 가지고 있다. 최근에 L2 캐쉬에 대한 흥미는 더욱더 높아 지고 있다. 3 세대로 알려 진 S/390 대칭다중처리장치의 초기판본에서 벌써 IBM 은 전용 L2 캐쉬를 리용하였다. 4 세대와 5 세대의 판본에서는 공유 L2 캐쉬가 리용되였다. 다음의 두가지 고찰은 이 변화를 보여 준다.

- 3 세대로부터 4 세대에로 발전하면서 극소형처리장치의 속도는 두배로 증가되였다. 만일 3세대가 유지되였다면 모선전송속도에서 커다란 변화가 일어 났을것이다. 같은 시기에 3 세대때의 구성요소들을 될수록 리용하려고 하였다. 모선에서 큰 발전이 없으면 모선교환망다리들이 난점으로 될것이다.
- 일반적인 S/390 의 작업량에 대한 분석은 처리장치들속에서 명령과 자료에 대한 큰 몫을 할당한다.

이상의 고찰에 기초하여 4 세대 S/390 설계집단은 매개 완충기가 하나이상의 L2 캐쉬를 리용하며 다중처리장치들에 의하여 공유되여야 한다는것을 제기하였다(그리고 매개 처리장치는 소편안에 L1 캐쉬를 전용으로 가지고 있다.). 얼핏보면 L2 하드웨어를 공유하는것이 잘못된것처럼 보인다. 그것은 처리장치가 기억기에 접근할 때 L2 캐쉬를 걸쳐야 하므로 더 속도가 떠지는것으로 보이기때문이다. 그런데 사실상 많은 량의 자료가 다중처리장치에 의하여 공유될 때 공유캐쉬가 있으면 그것을 방해하는것이 아니라 오히려 전체 처리량이 늘어 난다. 공유기억기안에서 공유되거나 검색되는 자료들은 모선을 통한

자료보다 훨씬 더 빨리 접근할수 있다.

4세대 S/390설계에 의하여 고려된 한가지 방법은 모든 처리장치들이 공유리용하는 하나의 큰 캐쉬문제이다. 한편 보다 캐쉬를 통하여 체계의 성능을 개선하려면 설계방법에서 현행체계모선조작을 완전히 다시 하지 않으면 안되게 된다. 그러나 성능해석이 보여주는바와 같이 현존 BSN 모선의 개개에 공유캐쉬를 도입하면 이에 의한 성능에서의 우점은 크게 나타나지만 한편으로는 모선통과량이 줄어 든다. 공유되는 캐쉬의 값은 성능측정에 의하여 확인되였는데 이것은 3 세대[MAK97]에 리용된 전용캐쉬방식보다 공유캐쉬가 훨씬 더 캐쉬명중률이 개선되였다는것을 보여 주었다[례 NAYF96].

L3(3 준위)캐쉬

S/390대칭다중처리장치체계의 다른 특징의 하나는 제3준위의 캐쉬(L3)를 쓴것이다[1]. L2캐쉬는 BSN에 배치되여 있으므로 개개의 L3캐쉬는 L2캐쉬와 기억기판사이의 완충을 보장한다. L3캐쉬는 접근하는 처리장치의 L1캐쉬와 L2캐쉬에 들어 있지 않는 자료에 대한 도달시간을 줄일수 있다. 이것은 접근된 캐쉬행이 다른 처리장치에 의하여 이미 공유되였지만 접근처리장치에 대해서는 아직 리용된적이 없었다면 주기억기보다 훨씬 더 빨리 자료를 접근할수 있다.

표 16-1은 심각한 기억 및 모선부하를 가진 표준 S/390의 작업부하에 따르는 3준위캐쉬로 장비된 대칭다중처리장치체계의 성능결과를 보여 주고 있다[DOET97][2]. 기억기접근위반은 캐쉬층에 대한 요구와 처음으로 복귀된 16bit 자료블로크사이의 지연시간이다. L1 캐쉬는 명중률이 89%이며 기억기참조의 나머지 11%는 L2, L3 과 주기억기준위들에서 보장된다. 11%가운데서 5%는 L2준위에서 해결된다. 3준위캐쉬를 리용함으로써 기억기참조는 불과 3%에 해당된다. 세번째 준위가 없으면 주기억기접근은 배로 늘어 나게 된다.

표 16-1. S/360 SMP 조직에서 캐쉬명중률

기억기보조체계	접근위반(PU 주기)	캐쉬크기	명중률(%)
L1 캐쉬	1	32KB	89
L2 캐쉬	5	256KB	5
L3 캐쉬	14	2MB	3
기억기	32	8GB	3

제 3 절. 캐쉬일치성과 MESI 규약

현대다중처리장치체계에서는 매개 처리장치와 련결된 하나 혹은 두개 준위의 캐쉬를 가지는것이 관례로 되여 있다. 이 구성은 본질상 성능을 높이기 위한것이다. 그러나 여기에서 제기되는것은 **캐쉬일치성**문제이다. 이 문제의 본질은 서로 다른 캐쉬에 같은

[1] IBM의 문헌들은 이 캐쉬를 L2.5캐쉬라고 한다. 사실상 이 캐쉬는 완충기의 세번째 준위로 구성하기때문에 이 용어로 특별한 우점이 없는것 같다.
[2] 자료는 설치된 L2캐쉬들은 리용하는 G3체계에 대한 것이다. 그러나 결과는 G4와 G5 S/390들에서 찾아볼수 있도록 공유 L2 캐쉬들에서 기대되는 성능에 대해 암시해준다.

자료를 동시에 여러개 복사할수 있거나 처리장치들이 자기의 캐쉬에 대한 복사를 자유롭게 갱신하는것을 허락하는 경우 기억기의 불일치성문제가 생길수 있다는것이다. 이미 제 4장에서 쓰기에 대한 가장 일반적인 두가지 방안을 정의하였다.

- **비동시쓰기(Write back)**: 쓰기조작은 오직 캐쉬에서만 진행된다. 주기억기는 대응하는 캐쉬의 행이 캐쉬로부터 지워 질 때만 갱신된다.
- **동시쓰기(Write through)**: 모든 쓰기조작은 캐쉬만이 아니라 주기억기에 대하여서도 진행된다. 여기서 주기억기는 항상 유효이라는것을 확인하여야 한다.

여기서 명백한것은 비동시쓰기방안에서 불일치성이 생길수 있다는것이다. 만일 두개의 캐쉬가 같은 행을 가지고 있고 그 행이 어느 한 캐쉬안에서 갱신되면 다른 캐쉬는 무효값을 가진다. 그다음의 읽기에서 무효인 행이 무효인 결과를 낳게 된다. 동시쓰기에서조차 다른 캐쉬가 기억기읽기쓰기를 하지 않거나 행에 대한 갱신통지를 직접 받지 못하면 불일치가 생길수 있다.

이 절에서는 캐쉬의 일치성문제에 대한 여러가지 해결방법을 간단히 고찰하고 그 가운데서 가장 널리 리용되는 MESI 규약에 대하여 집중적으로 고찰한다. 이 규약은 Pentium과 PowerPC를 실현하는데 리용되였다.

임의의 캐쉬일치성규약의 목적은 같은 시간에 여러개의 캐쉬안에 존재하는 공유변수의 일치성을 유지하기 위한 규약을 리용하는 한 가장 최근에 리용된 국부변수들이 여러차례의 읽기와 쓰기를 하여도 거기에 유지되게 하자는데 있다. 캐쉬일치성을 해결하는 방법에는 일반적으로 쏘프트웨어방법과 하드웨어방법이 있다. 일부 실현에서는 하드웨어와 쏘프트웨어요소들을 배합한 방법도 리용하고 있다. 쏘프트웨어방법과 하드웨어방법으로 분류하는것은 캐쉬일치성을 고찰하는데서 여전히 유익하면서도 흔히 리용되는 분류방법이다.

1. 쏘프트웨어해결방법

쏘프트웨어적으로 캐쉬일치성문제를 해결하기 위한 방안은 하드웨어회로들과 론리회로들을 더 보충함이 없이 이 문제를 콤파일러와 조작체계에 의거하여 취급하자는것이다. 쏘프트웨어방법은 잠재적인 문제들을 검사하는데 드는 전체 비용이 실행시간이 아니라 콤파일시간으로 반영되며 설계의 복잡성을 피하기 위하여 하드웨어가 아니라 쏘프트웨어로 한다는데 그 매력이 있다. 다른 한편 콤파일시간 즉 쏘프트웨어방법은 일반적으로 캐쉬를 리용함에 있어서 비효과적이라는 심각한 문제가 제기된다.

콤파일러에 기초한 일치성기구들은 어느 자료항목들을 보관할 때 불안전하게 될수 있겠는가를 결정하기 위하여 코드를 해석하고 그에 따라 그 항목들을 표식한다. 조작체계 혹은 하드웨어는 그때 보관할수 없는 항목들이 보관되는것을 금지시킨다.

제일 간단한 방법은 임의의 공유자료변수들이 보관되는것을 금지시키는것이다. 이것은 낡은 방법으로써 공유자료구조가 일정한 주기기간은 독점적으로 리용되며 다른 주기기간에는 효과적으로 읽기만을 할수 있기때문이다. 이것은 적어도 하나의 프로쎄스가 하나의 변수를 갱신하거나 적어도 다른 하나의 프로쎄스가 캐쉬일치성이 확인된 변수를 접근하는 기간에만 유효하다.

보다 효과적인 방법은 공유변수들에 대한 안전주기를 결정하기 위하여 코드를 해석하는것이다. 그때 콤파일러는 명령을 림계주기기간 캐쉬일치성을 실현하기 위해 발생시킨 코드에 삽입하는것이다. 해석을 진행하여 그 결과를 실행하기 위한 여러가지 수법들이 개발되고 있다. [LILJ93]과 [STEN90]을 참고하시오.

2. 하드웨어해결방법

하드웨어적인 해결방법을 일반적으로 캐쉬일치성규약이라고 한다. 이 해결방법은 잠재적인 불일치성조건들의 실행시간을 동적으로 인식하게 하는것이다. 문제가 실지로 발생했을 때만 취급되므로 보다 더 효과적으로 리용할수 있으며 결과적으로 쏘프트웨어방법보다 성능을 더 높일수 있다. 또한 이 방법은 프로그람작성자나 콤파일러가 쉽게 리해하므로 쏘프트웨어개발부담을 줄일수 있다.

하드웨어방법들은 자료행에 대한 상태정보를 어디에서 취하며 그 정보를 어떻게 구성하며 일치성을 어디에서 실현하며 그를 위한 기구를 어떻게 구성하는가 하는 일련의 문제에서 차이가 있다. 일반적으로 하드웨어방법은 등록부규약과 엿보기규약으로 분류된다.

등록부규약

등록부규약들은 행에 대한 복사가 준비되여 있는 장소에 대한 정보를 수집하고 유지하는것이다. 특히 여기에는 주기억조종기의 한 부분인 집중조종장치와 주기억안에 기억된 등록부가 있다. 등록부에는 여러가지 국부집중조종장치와 주기억안에 기억된 등록부가 있다. 등록부에는 여러가지 국부의 내용들에 대한 총체적인 상태정보가 들어 있다. 개별적인 캐쉬조종장치가 요구를 제기하면 집중조종장치는 기억기와 캐쉬들사이에서 혹은 캐쉬들자체의 사이에서 자료전송에 필요한 지령을 검사하고 출구한다. 이 장치는 또한 상태정보를 갱신할 때까지 유지하여야 할 의무를 지니고 있다. 따라서 행의 총체적인 상태에 영향을 줄수 있는 매개 국부동작상태를 중앙조종장치에 통보하여야 한다.

특히 조종장치는 처리장치가 어느 행을 복사하였는가에 대한 정보를 유지하고 있다. 처리장치는 행에 대한 국부복사를 하기전에 조종장치로부터 행에 대한 독점적접근을 요구하여야 한다. 이 독점적접근을 허락하기전에 조종장치는 이 행에 대한 캐쉬된 복사를 가지고 있는 모든 처리장치에 통보문을 보내며 매개 처리장치가 그 복사를 무효화하게 한다. 그러한 매개 처리장치로부터 다시 응답을 받은 다음 조종장치는 요구를 제기한 처리장치에 대하여 독점적인 접근을 허락한다. 다른 처리장치가 또 다른 처리장치에 독점적으로 허락된 행을 읽으려고 할 때에는 조종장치에 실패신호를 보내게 된다. 그 다음 조종장치는 그 행을 유지하는 처리장치에 지령을 출력하여 처리장치가 주기억기에 비동시쓰기를 하도록 한다. 그 행은 읽을 때 원천으로 되는 처리장치와 요구하는 처리장치에 의하여 공유될수 있다.

등록부규약들은 여러개의 캐쉬조종장치와 중앙조종장치사이에 통신파잉과 같은 난점을 초래한다. 그러나 이 규약들은 다중모선체계나 다른 복잡한 호상련결체계를 가진 대규모체계에서 매우 효과적이다.

엿보기규약

엿보기규약은 다중처리장치안의 모든 캐쉬장치에서 캐쉬일치성을 유지하기 위한 작업을 분산하는 규약이다. 캐쉬는 자기가 취한 행이 다른 캐쉬와 공유되여 있다는것을 인식하여야 한다. 공유캐쉬의 행에 대한 갱신이 진행되였을 때 통보기구에 의하여 다른 모든 캐쉬에 그것을 알려 주어야 한다. 매개 캐쉬조종장치들은 알려 주는 통지를 감시하며 그에 따라 즉시에 반응하기 위하여 련결망에 대한 **엿보기**를 진행한다.

엿보기규약은 리상적으로 모선에 기초한 다중처리장치체계에 맞는다. 왜냐하면 공유모선이 통지와 엿보기하는데서 제일 단순한 수단으로 되기때문이다. 그러나 국부캐쉬를 리용하는 목적의 하나는 모선접근을 피하는것과 통지와 엿보기를 위하여 요구되는 모선에 대한 통신량의 증대가 국부캐쉬의 리용으로 하여 얻어 지는 리득을 떨구지 않게 하자는것이다.

엿보기규약에 대하여 두가지 기본방법인 쓰기-무효규약과 쓰기-갱신(혹은 쓰기-통보)규약이 있다. 쓰기-무효규약에서는 읽는 측은 여럿일수 있지만 쓰는 측은 다만 하나이다. 처음에 읽기를 위하여 한행을 여러개의 캐쉬들이 공유한다. 캐쉬들가운데서 어느 하나가 행에 대한 쓰기를 하려고 할 때 먼저 다른 캐쉬안의 그 행이 무효라는 통지를 보내여 쓰기하는 캐쉬가 독점적이 되게 한다.

일단 그 행이 독점적이 되면 처리장치는 다른 처리장치들이 같은 행을 요구할 때까지 국부적인 쓰기를 한다.

쓰기-갱신규약에서는 여러개의 쓰기 혹은 읽기측이 있을수 있다. 처리장치가 공유된 행을 갱신하려고 할 때 갱신하려는 단어는 다른 모든 처리장치들에 분배되며 캐쉬들은 그에 대한 갱신을 계속한다.

이 두 방안은 모든 환경에서 다른 방안들에 비하여 다 우수한것은 아니다. 성능은 국부캐쉬의 수와 기억기읽기와 쓰기의 패턴에 관계된다. 일부 체계들은 쓰기-무효와 쓰기-갱신의 과정을 둘다 가진 적응형규약들을 리용하고 있다.

쓰기-무효방식은 PentiumⅡ, PowerPC와 같은 상품화된 다중처리장치체계에서 가장 널리 리용되고 있다. 여기서는 캐쉬꼬리표에 비트를 더 추가하여 변경, 독점, 공유, 무효에 따라 매개 캐쉬행의 상태를 표시한다. 이러한 리유로 하여 쓰기-무효규약을 MESI라고 부른다. 이미 제4장에서 1준위와 2준위 국부캐쉬사이의 조종을 취급하면서 MESI에 대하여 보았다. 이 절의 뒤에서는 다중처리장치를 통하여 국부캐쉬안에서 이 규약의 리용을 고찰하기로 한다. 설명을 간단히 하기 위하여 같은 시간안에 분산된 다중처리장치들을 통한 조종과 동시에 1준위와 2준위를 둘 다 국부적으로 조종하는것과 같은 기구는 취급하지 않는다. 이것은 그 어떤 새로운 원리는 아니지만 론의에서 복잡해 질수 있기때문이다.

3. MESI규약

MESI 규약에서 자료캐쉬는 매개 행이 4개 상태가운데서 어느 한 상태에 있도록 매개 꼬리표에 두개의 상태비트를 할당하고 있다는것을 제4장에서 보여 주었다.

- **변경**: 캐쉬안의 행은 주기억과는 달리 변경되며 이것은 이 캐쉬에서만 가능하다.

- **독점**: 캐쉬안의 행은 주기억기안의것과 같으며 그 어떤 다른 캐쉬안에는 존재하지 않는다.
- **공유**: 캐쉬안의 행은 주기억기안의것과 같으며 다른 캐쉬안에도 존재할수 있다.
- **무효**: 캐쉬안의 행은 유효자료를 가지지 않는다.

그림 16-8 에 MESI 규약에 대한 상태도를 보여 주었다. 기억해 두어야 할것은 캐쉬의 매개 행은 자기자체의 상태비트를 가지고 있으며 상태도를 자체로 실현할수 있다는것이다. 그림 16-8 ㄱ에 이 캐쉬에 접근한 처리장치에 의하여 생긴 동작으로 인한 상태전이를 보여 주었다. 그림 16-8 ㄴ에 공통모선에서 엿보기되는 사건에 의하여 일어 나는 상태전이를 보여 주었다. 처리장치—발화동작과 모선초기화동작을 개별적상태로 표현하는것은 MESI 규약의 론리적과정을 명백히 하는데 도움을 준다. 그러나 어떤 때는 캐쉬가 단일상태에 있는 경우가 있다. 만일 다음사건이 접근된 처리장치로부터 제기되였다면 그때 전이는 그림 16-8 ㄱ에서와 같이 되며 모선으로부터 제기되였다면 전이는 그림 16-8 ㄴ와 같이 된다. 이 전이과정을 좀 더 구체적으로 보기로 하자.

읽기실패

국부캐쉬에 대한 읽기실패가 발생하였다면 처리장치는 실패한 주소가 있는 주기억기의 행을 읽기 위하여 기억읽기를 시작한다. 처리장치는 조종신호를 모선에 내보내여 다른 모선처리장치들과 캐쉬들이 처리요구(transaction)를 엿보기할수 있게 한다. 몇가지 가능한 경우는 다음과 같다.

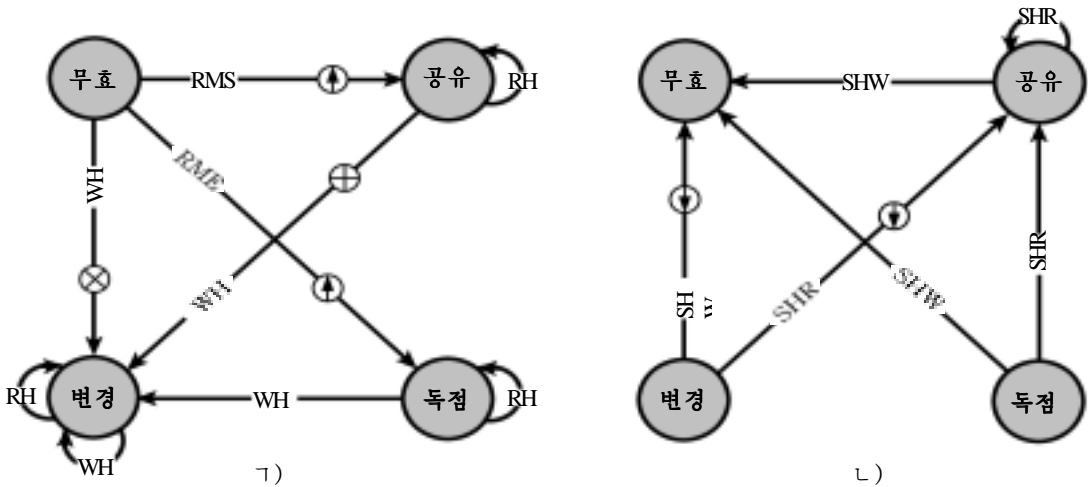



그림 16-8. MESI 상태동작
ㄱ-기동처리장치에서 캐쉬의 행, ㄴ-전용캐쉬의 행

- 다른 캐쉬가 독점적상태에서 행에 대한 복사가 명백하면(기억기로부터 읽기한 후에 변경이 없었다면) 이 행을 공유했다는것을 가리키는 신호는 복귀된다. 그 다음 응답하는 처리장치는 독점적상태로부터 공유상태로 자기의 복사상태를 전이하며 발화하는 처리장치는 주기억기로부터 행에 대한 읽기를 하고 그 캐쉬안의 행을 무효상태로부터 공유상태로 전이한다.
- 하나이상의 캐쉬공유상태에서 행에 대한 복사가 명백하면 그들의 개개는 모선을 공유했다는 신호를 내보낸다. 발화하는 처리장치는 그 행을 읽으며 그 캐쉬안의 행은 무효상태로부터 공유상태에로 전이한다.
- 하나의 다른 캐쉬가 행에 대한 복사를 변경하였다면 그때 그 캐쉬는 읽기를 차단하고 공유모선을 통하여 요구하는 캐쉬에 행을 제공한다. 그 다음 응답하는 캐쉬는 그 행을 변경상태로부터 공유상태로 바꾼다.[3]
- 다른 캐쉬가 행을 복사하지 않고 있으면 그때 신호는 복귀되지 않는다. 발화하는 처리장치는 행을 읽고 [3]그 캐쉬의 행을 무효상태로부터 독점상태에로 전이한다.

읽기명중

국부캐쉬에서 현재 행에 대한 읽기가 명중하였다면 처리장치는 필요한 항목을 간단히 읽는다. 상태변화가 없으면 변경, 공유 그리고 독점상태가 유지된다.

쓰기실패

국부캐쉬에서 쓰기가 실패하였을 때 처리장치는 실패한 주소가 들어 있는 주기억기의 행을 읽기 위하여 기억기읽기조작을 시작한다. 이를 위하여 처리장치는 ≪변경을 하려는 읽기≫라는 의미에서 RWITM 이라는 신호로 모선에 내보낸다. 행이 넣어 졌을 때 그것은 즉시에 변경을 표식한다. 다른 캐쉬에 대하여 자료의 행에 넣기를 위한 두가지 방법론이 준비되고 있다.

첫번째 방법론은 일부 다른 캐쉬가 이 행에 대한 변경된 복사를 한다. 이 경우 처리장치는 다른 처리장치가 행에 대한 변경된 복사를 한다는것을 발화처리장치에 알린다. 발화처리장치는 모선을 넘겨 주고 대기한다. 다른 처리장치는 모선에 대한 접근권을 얻어 변경된 캐쉬행을 도로 주기억기에 써넣으며 캐쉬행의 상태를 무효상태로 전이한다. (그것은 발화처리장치가 이 행을 변경시키고 있기때문이다.) 다음으로 발화하는 처리장치는 다시 RWITM 의 모선상태신호를 출력하고 그다음 주기억기로부터 행을 읽으며 캐쉬의 행을 변경하고 변경된 상태를 행에 표시하는 단계를 차례로 수행한다.

두번째 방법론은 요구되는 행에 대해 복사를 변경하지 않는것이다. 이 경우에 신호는 복귀되지 않으며 발화된 처리장치는 행안에서 읽기를 진행하고 그것을 변경한다. 이 기간 하나 혹은 그이상의 캐쉬가 공유상태에 있으면서 행에 대한 복사가 명백하다면 매개 캐쉬는 그 행에 대한 복사를 무효화한다. 하나의 캐쉬가 독점적상태에서 행에 대한 복사가 명백하다면 행에 대한 복사는 무효화된다.

---

[3]일부 실행들로하여 캐쉬는 변경상태의 행신호들을 초기화처리장치에서 진행한다. 한편 처리장치는 변경상태의 복사가 모선을 리용함으로하여 주기억기다음상태에서 변경상태의 행쓰기를 하며 변경상태로부터 공유상태의 캐쉬에 행을 전이한다. 점차 처리장치는 재실행을 요구하며 하나 혹은 그이상의 처리장치가 앞서 말한바와 같이 공유상태에서 행복사를 진행한것을 탐색한다.

쓰기명중

국부캐쉬안에서 현재 행에 대한 쓰기가 명중했다면 그 효과는 국부캐쉬안의 그 행의 현재상태에 관계된다.

- **공유:** 갱신이 진행되기전에 처리장치는 행에 대한 독점적소유권을 얻어야 한다. 처리장치는 모선에 그를 위한 신호를 보낸다. 그 캐쉬안에서 행에 대한 복사가 공유된 매개 처리장치들은 공유상태로부터 무효상태에로 상태전이를 한다. 이때 발화된 처리장치는 갱신을 진행하며 공유상태로부터 변경상태에로 행에 대한 복사를 전이한다.
- **독점:** 처리장치는 이미 이 행에 대한 독점적조종상태에 있으므로 갱신을 간단히 수행하며 행에 대한 복사를 독점적인 상태로부터 변형된 상태에로 전이한다.
- **변경:** 처리장치는 이미 이 행에 대한 독점적조종상태에 있으므로 그 모선은 변경상태로 표식되여 있으며 갱신을 간단히 수행할수 있다.

1 준위(L1)-2 준위(L2) 캐쉬일치

이미 같은 모선이나 다른 대칭다중처리장치체계의 호상련결기구에 련결된 캐쉬들사이의 호상협조동작의 견지에서 캐쉬일치성규약을 서술하였다. 특히 이 캐쉬들은 2 준위(L2)캐쉬이며 매개 처리장치는 또한 모선에 직접 련결하지 않으며 따라서 엿보기규약에 들어 있지 않는 1 준위(L1)캐쉬를 가지고 있다. 따라서 일부 방안들은 두 준위의 캐쉬를 통하여 그리고 대칭다중처리장치조직안의 모든 캐쉬들을 통하여 자료를 보존하여야 한다.

중요한것은 MESI 규약(혹은 임의의 일치성규약)을 L1 캐쉬에까지 확장하는것이다. 따라서 L1 캐쉬안의 모든 행은 상태를 표현하는 비트들을 포함하고 있다. 그 목적은 다음과 같다. L2 캐쉬와 그에 대응하는 L1 캐쉬안에 있는 임의의 행에 대하여 L1 행의 상태는 L2 행의 상태를 따른다. 이를 위한 간단한 방법은 L1 에 동시쓰기방식을 적용하는것이다. 이 경우에 동시쓰기는 기억기가 아니라 L2 캐쉬에 한다. L1 의 동시쓰기방식은 L2 캐쉬가 아니라 L1 행을 변경시키므로 이것을 다른 L2 에서 볼수 있다. L1 에 동시쓰기방식을 리용할 때 L1 내용은 L2 내용의 일부분이여야 한다. 따라서 이것은 L2 캐쉬의 련상성이 L1 의 련상성(Associativity)과 같거나 더 커야 한다는것을 말해 준다. L1 동시쓰기방식은 IBM S/390 콤퓨터에서 리용되였다.

만일 L1 캐쉬가 비동시쓰기방식을 리용한다면 두 캐쉬사이의 관계는 더 복잡해 진다. 일치성을 유지하기 위한 여러가지 방안들이 제기되고 있는데 Pentium Ⅱ에서 리용된 방식은 참고문헌[SHAN98]에 구체적으로 서술되여 있다.

제 4 절. 클라스터

최근 콤퓨터체계설계에서 가장 본격화되고 있는것은 클라스터화이다. 클라스터화(콤퓨터병렬화)는 고성능과 높은 리용률을 얻기 위하여 대칭다중처리방식으로 나가고 있으며 봉사기적용에 특별히 적중하다. 클라스터는 단일한 계산자원을 함께 처리하는 호상련결된 **전체 콤퓨터**들의 하나의 그룹이라고 정의할수 있다. 전체 콤퓨터라는 말은 개별

적콤퓨터들이 클라스터와 독립적으로 실행할수 있는 하나의 체계를 의미한다. 문헌에서는 이 병렬콤퓨터의 매개 콤퓨터들을 **마디**라고 부르고 있다.

문헌[BEEW97]에서는 콤퓨터의 병렬화로 얻어 지는 우점을 4 가지로 분류하였다. 이것들은 설계요구 혹은 설계목표로 볼수 있다.

- **절대스칼라성:** 제일 큰 자체봉사기능을 갖춘 기계의 능력을 훨씬 초과하는 대규모클라스터를 실현할수 있다.
- **확장스칼라성:** 병렬콤퓨터는 새로운 체계를 조금씩 보충할수 있게 구성된다. 따라서 사용자는 자기에게 알맞는 체계로부터 시작하여 현재의 작은 체계를 보다 큰 체계로 교체함이 없이 필요에 따라 그것을 확장한다.
- **높은 리용률:** 병렬콤퓨터의 매개 마디는 독립콤퓨터로 되여 있으며 따라서 하나의 마디에서 오유가 생긴다고 하여 봉사에서 손실을 본다고 말할수 없다. 많은 제품들에서 고장은 쏘프트웨어에 의하여 자동적으로 조종된다.
- **높은 가격 대 성능비:** 상품화된 개별적콤퓨터들을 결합하여 구성하므로 하나의 큰 기계와 같거나 그보다 더 높은 처리성능을 가지면서 훨씬 눅은 병렬콤퓨터를 만들수 있다.

1. 병렬콤퓨터의 조직

문헌들에서 병렬콤퓨터는 여러가지 방법으로 분류하고 있다. 제일 간단한 분류는 병렬콤퓨터안의 콤퓨터들이 같은 디스크에 대한 접근을 공유하는가 안하는가에 기초한다. 그림 16-9 ㄱ에 클라스터의 동작을 조종하기 위한 통지문교환에 리용되는 고속련결로에 의하여서만 호상 결합된 두마디병렬콤퓨터를 보여 주었다. 련결기구는 다른 비클라스터콤퓨터와 공유되는 국부망일수도 있으며 전용호상련결기구일수도 있다. 후자의 경우에 클라스터안의 하나 혹은 여러개의 콤퓨터는 봉사기클라스터와 원격의뢰기체계사이의 련결을 보장하도록 국부망이나, 광대역망에 접속되여야 한다. 주의할것은 그림에서 매개 콤퓨터들은 다중처리장치체계로 되여 있는것처럼 그려져 있다는것이다. 이것은 필요 없을뿐아니라 성능과 리용가치도 높지 못하다.

그림 16-9 에서 보여 준 간단한 분류에서 다른 하나의 방식은 디스크공유클라스터이다. 이 경우에 일반적으로 마디들사이의 통지문통로는 여전히 존재한다. 게다가 여기에는 클라스터안의 다중콤퓨터들에 직접 련결되는 디스크보조체계가 있다. 이 그림에서 공동디스크보조체계는 RAID 체계이다. RAID 나 그와 류사한 일부 여유디스크기술은 다중콤퓨터에 의하여 실현되는 높은 리용가치가 하나의 오유가 있는 공유디스크에 의하여 떨어지지 않도록 하기 위하여 리용되고 있다.

클라스터선택의 범위에 대한 보다 명백한 구상은 기능선택에 의하여 얻어 진다. 휼래트-팩카드[HP96]의 백서에는 기능에 따르는 분류가 소개되여 있다(표 16-2).

일반적이면서도 낡은 방법인 **피동대기**(passive standby)방법은 단순히 다른 콤퓨터가 비능동상태인 경우에 한 콤퓨터가 모든 처리부하를 담당하는것으로서 이 처리부하는 1 차측의 고장으로 하여 대기하고 있다. 능동이면서 1 차인 콤퓨터를 조종하기 위하여 체계는 대기하는 기계에 《동작중》의 통보문을 계속 보낸다. 이 통보문이 수신되지 않으면 대기중의 기계는 1 차봉사기가 고장났다는것을 예측하고 자기가 동작상태로 들어 간다.

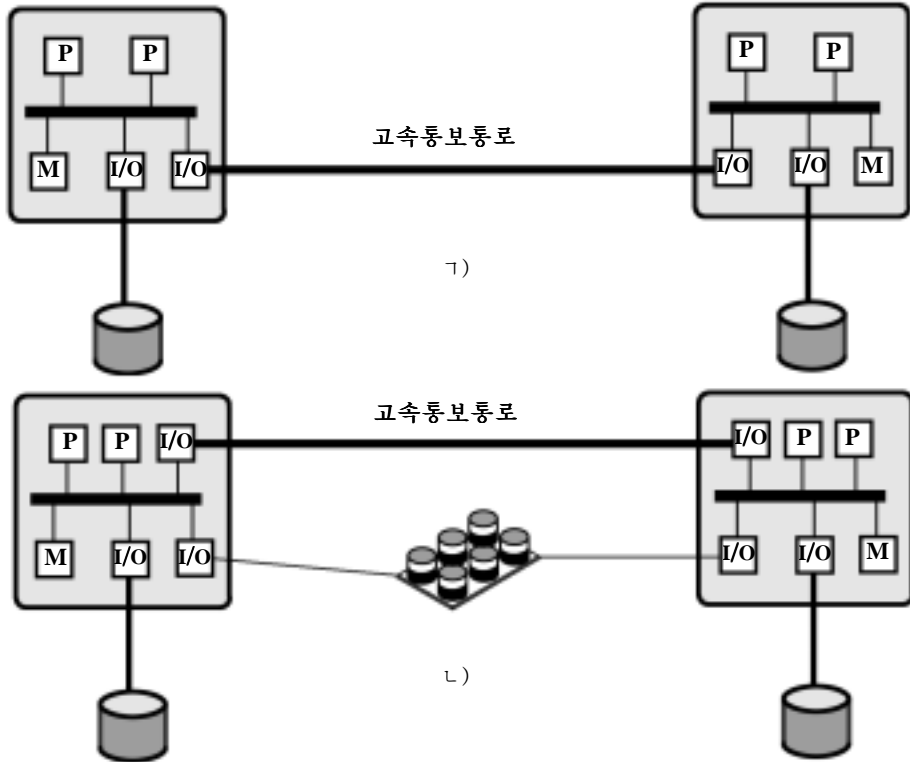



그림 16-9. 클라스터의 구성

ㄱ- 비공유디스크를 가진 스텐바이봉사기, ㄴ- 공유디스크

이 방안은 리용가치는 있지만 성능을 개선할수 없다. 더우기 두 체계사이에 교환되는 정보가 《동작중》상태라는것밖에 없고 두 체계가 공동디스크를 공유하지 않는다면 그때 대기하는 측은 기능지원을 하지만 1차측에 의하여 관리되는 자료에 접근하지 못한다.

피동대기방법은 일반적으로 클라스터에서 쓰이지 않는다. **클라스터**라는 말은 호상련결된 콤퓨터를 념두에 둔것으로서 모든 처리장치들이 능동적으로 처리하지만 외적으로는 마치도 하나의 콤퓨터와 같이 보인다. 이 구성에서는 흔히 **능동2차방법**을 쓴다. 병렬화는 세가지 즉 개별봉사기들, 비공유 및 공유기억기로 분류할수 있다.

병렬화의 첫번째는 개별봉사기방식인데 이것은 매개 콤퓨터가 **분리된 봉사기**로서 이 봉사기들은 자기자체의 디스크는 가지고 있지만 체계들사이에 공유하는 디스크는 없다(그림 16-9 ㄱ). 이러한 배렬은 높은 성능과 리용가치를 가진다. 이 경우 부하의 균형을 맞추고 높은 리용률을 보장하기 위하여 입력되는 의뢰기의 요구들을 봉사기에 할당하려면 몇가지 관리형태와 쏘프트웨어의 순서짜기가 요구된다. 이것은 고장극복능력을 가지도록 하자는것으로서 콤퓨터가 응용프로그람을 실행하는 도중에 고장이 생기면 클라스터안의 다른 콤퓨터가 그것을 장악하여 응용프로그람의 실행을 완료한다. 이를 위하여 자료는 늘 체계안에 복사되여 있어야 하며 이렇게 하여야 매개 체계가 다른 체계의 현행자료를 접근할수 있다. 이 자료교환을 위한 보조조작은 성능에서 손실은 보지만 높은 리용가치를 보장한다.

표 16-2. 집합방법: 우점과 제한성

집합방법	설명	우점	제한성
피동대기	1 차봉사기가 고장인 경우 2 차봉사기가 기동한다	실현하기 쉽다	2 차봉사기를 다른파제처리에 리용할수 없으므로 비용이 높다.
2 차능동:	2 차봉사기도 파제처리에 리용한다.	2 차봉사기를 파제처리에 리용할수 있으므로 비용이 낮다.	복잡성이 증가한다.
봉사기분리	분리된 봉사기들이 자기의 디스크를 가진다. 자료가 1 차에서 2 차봉사기로 련속적으로 복사된다.	높은 리용성을 가진다.	조작복사로 인한 망 및 봉사기간접소비시간이 길어진다.
봉사기를 디스크에 련결	봉사기를 같은 디스크에 련결하지만 매개 봉사기는 자기의 디스크를 가진다. 어떤 봉사기가 고장나면 다른 봉사기가 그의 디스크들과 련결된다.	조작복사가 없으므로 망 및 봉사기간접소비시간이 단축된다.	디스크고장 위험을 보상하기 위하여 보통 디스크미러링(mirroring) 혹은 RAID 기술을 요구한다.
봉사기들이 디스크를 공유	여러개의 봉사기들이 디스크에 대한 접근을 동시에 진행한다.	망 및 봉사기간접소비시간이 짧다. 디스크고장으로 인한 수리시간이 감소된다.	관건관리기쏘프트웨어를 요구한다. 보통 디스크미러링 혹은 RAID 기술을 함께 리용한다.

통신에서 이 보조조작을 줄이기 위하여 대부분의 클라스터는 현재공동디스크에 련결된 봉사기들로 구성한다(그림 16-9 ㄴ). 이 방식가운데서 하나의 변종을 간단히 **공유없는 방식**이라고 한다. 이 방식에서 공동디스크는 용량별로 나누어 져 있으며 매개 용량은 개별적콤퓨터에 의하여 관리된다. 만일 그 콤퓨터가 고장이면 클라스터는 다른 콤퓨터들이 고장난 콤퓨터의 용량에 대한 소유권을 가지도록 재구성되여야 한다.

또한 다중콤퓨터들은 같은 시간에 같은 디스크를 공유할수 있어야 하는데(이것을 **공유디스크방식**이라고 한다.) 그렇게 하여야 매개 콤퓨터가 모든 디스크에 있는 용량들을 접근할수 있다. 이 방법은 자료가 한번에 하나의 콤퓨터에 의해서만 접근될수 있다는것을 확인하기 위하여 몇가지 류형의 관건기구를 리용하는것이 필요하다.

2. 조작체계설계항목

클라스터의 하드웨어구성을 충분히 리용하기 위하여서는 단일처리장치체계의 조작체계에 다음과 같은 몇가지 기능을 더 첨부한다.

고장관리

집합화하는 방법에 따르는 클라스터에 의하여 고장은 어떻게 관리되는가(표 16-2). 일반적으로 고장을 처리하기 위하여 두가지 방법 즉 클라스터방식과 고장허용(내고장성) 클라스터방식이 있다. 클라스터는 모든 자원들이 복사중에 있어야 할 확률이 높다. 만일 체계가 정지되거나 디스크용량을 잃어 버리는것과 같은 고장이 일어 나면 차례로 질문을 잃어 버린다. 다시말하여 클라스터안의 서로 다른 콤퓨터들이 임의의 잃어 버린 질문들을 봉사한다. 그러나 클라스터의 조작체계들은 부분적으로 실행된 처리요구의 상태에 대

하여 보증하지 않는다. 이것은 응용프로그람준위에서 조종되는것이 필요하다.

고장허용클라스터는 모든 자원을 쓸수 있게 확보한다. 이것은 여유공유디스크를 리용하든가 해당되지 않는 처리요구를 취소하는 기구를 쓰든가 완성된 처리요구를 위탁하든가 하는 방법으로 해결한다.

고장난 체계로부터 그 클라스터안의 선택된 체계에로 응용프로그람과 자료자원을 절환하는 기능을 **고장극복**이라고 한다. 그와 관계되는 기능을 일단 그것이 고착되면 본래체계에로 응용프로그람의 자료를 복귀한다. 이것을 **고장복귀**라고 한다. 고장복귀는 자동적으로 할수 있지만 이것은 그 문제가 정확히 고착되였거나 복귀하는것이 명백한 경우에만 가능하다. 만일 그렇지 않으면 자동적인 고장복귀는 고장난 자원들이 콤퓨터들사이에서 오락가락하게 만들어 도리여 성능과 복귀에 문제를 야기시킨다.

부하균형

클라스터는 가능한 콤퓨터들안에서 부하균형을 맞추기 위한 능력을 가져야 한다. 여기에는 클라스터를 점차적으로 스칼라화하려는 요구가 담겨져 있다. 새로운 콤퓨터가 클라스터로 확장될 때 부하균형기구는 응용프로그람의 순서를 짜면서 이 콤퓨터에 자동적으로 포함된다. 미들웨어(middleware)기구는 클라스터의 서로 다른 개별콤퓨터에 봉사가 나타 날수 있으며 한 개별콤퓨터로부터 다른 개별콤퓨터에로 옮겨 갈수 있다는것을 인식해야 한다.

3. 클라스터와 대칭다중처리장치콤퓨터

클라스터들과 대칭다중처리장치콤퓨터들은 둘 다 높은 요구를 제기하는 응용프로그람을 지원하는 다중처리장치들로 구성되여 있다. 이 두 방식은 상업적으로 가능하다.

대칭다중처리장치방식의 기본우점은 클라스터방식보다 관리와 구성이 쉽다는것이다. 이 방식에서는 거의 모든 응용프로그람작성모형이 단일처리장치방식과 밀접한 관계가 있다. 단일처리장치로부터 대칭다중처리장치로 전환되는데서 요구되는 기본요인은 순서짜기기능이다. 대칭다중처리장치콤퓨터의 다른 우점은 일반적으로 클라스터보다 물리적공간이 작고 전력소비가 적은것이다. 대칭다중처리장치의 가장 중요한 또 하나의 우점은 설치가 쉽고 안정한것이다.

그런데 집합콤퓨터방식의 우점은 고성능봉사기시장을 이 콤퓨터들이 지배하는것이다. 클라스터들은 증분과 절대스칼라성의 견지에서 보면 대칭다중처리장치방식보다 훨씬 더 앞서 있다. 클라스터들은 체계의 모든 요소들이 높은 여유성을 보장할수 있기때문에 리용성의 견지에서도 우월하다.

제 5 절. 비균일기억기접근

상품의 견지에서 응용프로그람을 지원하기 위하여 다중처리장치체계를 갖춘 두가지 대표적인 방식은 대칭다중처리장치방식과 클라스터방식이다. 몇해동안 비균일기억기접근이라고 불리우는 새로운 방법이 연구대상으로 되고 있으며 최근에는 이것을 리용한 제품들이 나타나고 있다.

선행에 앞서서 비균일기억기접근에 관한 문헌들에서 나오는 몇가지 용어를 정의하기로 하자.

- **비균일기억기접근(UMA):** 모든 처리장치들은 읽기와 쓰기를 리용하여 주기억기의 모든 부분을 접근할수 있다. 기억기의 모든 부분에 대한 접근시간은 꼭 같다. 제2절과 제3절에서 론의한 대칭다중처리장치체계가 이 방식에 속한다.
- **비균일기억기접근(NUMA):** 모든 처리장치들은 읽기와 쓰기를 리용하여 주기억기의 모든 부분을 접근할수 있다. 처리장치의 기억기에 대한 접근시간은 주기억령역의 어느곳을 접근하는가에 따라 다르다. 서로 다른 처리장치에 있어서 어떤 기억령역에서는 접근이 더 빨라 지고 어떤 기억령역에 대해서는 접근이 더 떠진다.
- **캐쉬일치성비균일기억기접근(CC-NUMA):** 캐쉬의 일치성이 여러가지 처리장치의 캐쉬안에서 유지되는 비균일기억기접근체계를 말한다.

캐쉬일치성이 없는 비균일기억기접근체계는 어느 정도 클라스터체계와 등가이다. 최근에 주목을 끄는 상품들은 캐쉬일치성이 있는 비균일기억기접근체계들이며 대칭다중처리장치체계와는 완전히 구별된다. 이러한 체계는 문헌에서 캐쉬일치성기능을 가진 비균일기억기접근체계라고 한다. 이 절에서는 이 체계에 대해서만 론의한다.

1. 비균일기억기접근을 리용하게 된 동기

대칭다중처리장치체계에는 리용할수 있는 처리장치의 수에 실천적인 제한성이 있다. 캐쉬의 리용에서 효과성이 높아 지는데 따라 어느 한 처리장치와 주기억기상의 모선전송능력은 감소한다. 처리장치의 수가 증가함에 따라 이 모선전송능력은 증가한다. 모선은 또한 캐쉬의 일치성신호들을 교환하는데도 리용되므로 모선부하는 더 커진다. 일부 측면에서 모선은 성능을 높이는데 난점으로 된다. 성능의 감소는 대칭다중처리장치조직에서 16개로부터 64개 범위로 처리장치의 수를 제한한다. 례하면 실리콘그라픽스파워첼린져회사의 대칭다중처리장치콤퓨터는 하나의 체계에 R10000처리장치가 64개로 제한되여 있다. 이 수를 넘으면 성능은 실제로 감소한다.

대칭다중처리장치체계에서 처리장치수의 제한은 집합처리장치체계의 개발을 추동한 요인의 하나이다. 그런데 집합처리장치체계에서 매개 마디는 자기자체를 위한 국부주기억기들을 따로 가지고 있으므로 응용프로그람들은 큰 공동기억기를 요구하지 않는다. 실제로 일치성은 장치에 의해서가 아니라 쏘프트웨어에 의하여 유지된다. 이 기억기접근은 성능에 영향을 주며 최대의 성능을 얻기 위하여 쏘프트웨어가 이 환경에 맞아야 한다. 대칭다중처리장치체계의 우점을 유지하면서 대규모다중처리를 실현하기 위한 한가지 방법은 비균일기억기접근을 실현하는것이다. 례하면 실리콘그라픽스오리진회사의 비균일기억기접근체계는 MIPS R10000 처리장치를 1024개까지, 시켄드 NUMA-Q 체계는 Pentium Ⅱ 처리장치를 252개까지 지원할수 있게 설계되였다[LOVE96].

비균일기억기접근의 목적은 매개 마디가 개별적인 모선 혹은 다른 내부호상결합체계를 가지는 여러개의 다중처리장치마디들을 허가하는 한편 현존하는 체계의 넓은 대역기억기를 가지도록 하는것이다.

2. 내부조직

그림 16-10에서는 전형적인 CC-NUMA의 내부조직을 보여 주었다. 여기에는 여러개의 독립적인 마디들이 있는데 이 매개 마디가 실제로 대칭다중처리소방식으로 되여

있다. 따라서 매개 마디는 다중처리장치들로 되여 있으며 그 개개 처리장치들은 자체의 L1 과 L2 캐쉬와 주기억기를 가지고 있다. 마디는 완전한 CC-NUMA 로 구성되여 있다. 례하면 실리콘그라픽스오리진회사의 NUMA 체계의 매개 마디는 두개의 MIPS R10000 처리장치로 되여 있으며 NUMA-Q 회사의 마디들은 Pentium II 처리장치 4 개로 구성되여 있다. 마디들은 여러가지 통신기구 즉 교환기구, 고리 혹은 몇가지 다른 망을 리용하여 호상 련결되여 있다.

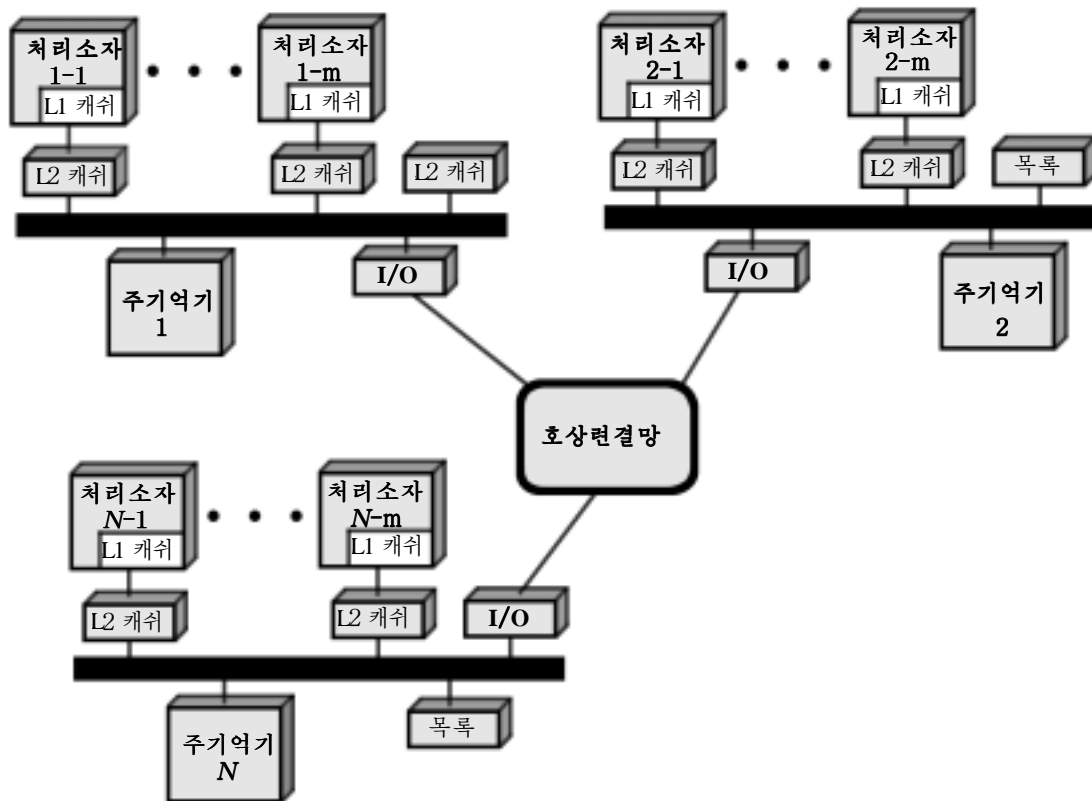



그림 16-10. 캐쉬일치성비단조기억접근구성

통신 즉 CC-NUMA 체계안의 매개 마디는 일부 주기억들을 가지고 있다. 그러나 처리장치의 견지에서 보면 주소화할수 있는 기억기를 하나만 가지고 있는데 그 기억기는 각각 단일체계범위의 주소를 관찰할수 있는 국부기억기이다. 처리장치가 기억기접근을 시작할 때 접근단기억주소가 처리장치의 캐쉬안에 존재하지 않는다면 그때 L2 캐쉬는 꺼내기조작을 한다. 만일 요구되는 행이 주기억기안의 국부적인 분류에 있다면 그 행을 국부모선을 통하여 꺼내게 된다. 요구되는 행이 주기억기에서 멀리 떨어 져 있다면 호상결합망을 통하여 그 행을 꺼내기 위한 자동요구를 내보내며 그것을 국부모선을 거쳐 그다음 그 모선에 련결된 요구하는 캐쉬에 전달한다. 이 모든 동작은 처리장치와 그의 캐쉬에서 자동적으로 진행된다.

이 구성에서 캐쉬일치성문제가 중심을 이룬다. 비록 실현함에 있어서 구체적인 측면에서 차이가 있지만 일반적으로 매 마디는 기억기의 여러 부분의 주소와 캐쉬의 상태정보를 지적하는 몇종류의 목록을 유지하고 있어야 한다는것이다. 이 방안이 어떻게 동작

하는가를 보기 위하여 실례를 [PEIS98]에 주었다. 마디 3의 처리장치 2(P2-3)은 기억주소 798을 요구하는데 이 주소가 마디안의 기억주소를 의미한다면 다음과 같이 동작과정에 있게 된다.

- P2-3은 주소 798을 접근하기 위하여 마디 2의 엿보기모선에 읽기요구를 출력한다.
- 마디 2에 대한 목록은 그 요구를 통하여 주소가 마디 1안에 있다는것을 인식한다.
- 마디 2의 목록은 마디 1에 요구를 내보내고 그리고 마디 1의 목록에 의하여 선택된다.
- P2-3을 대신하는 마디 1은 마치도 그것이 처리장치인것처럼 798주소의 내용을 요구한다.
- 마디 1의 주기억은 모선우에 요구하는 자료를 태웠음을 응답한다.
- 마디 1의 목록은 모선으로부터 자료를 선택한다.
- 그값은 마디 2의 목록에로 도로 전송된다.
- 마디 2의 부록은 그것을 처음에 취했던 기억기를 대신하는 마디 2의 모선에 자료를 다시 놓는다.
- 값은 선택되여 P2-3의 캐쉬에 기억되여 다시 P2-3을 읽는다.

다음으로 처리요구가 처리장치에 쏘프트웨어기구를 리용하여 원격기억기로부터 자료를 어떻게 읽는가 하는것을 보기로 하자. 이 기구에 몇가지 형태의 캐쉬일치성규약이 필요하다. 여러가지 체계에서는 이것을 어떻게 하는가 하는데 차이가 있다. 여기서 일반적인 몇가지 문제에 대해서만 보기로 하자. 먼저 선행과정의 한 부분으로서 마디 1의 목록은 일부 원격캐쉬가 주소 798안에 있는 항을 복사한 기록을 가지고 있다. 그때 변경을 위하여 협동규약이 필요하다. 례하면 변경이 캐쉬안에서 집행되면 이 사실은 다른 마디들에 통보된다. 그러한 통보를 받는 매개 마디의 목록은 임의의 국부가 그 행을 가지고 있겠는가 그것을 삭제하였는가를 결정할수 있다. 만일 실제의 기억주소가 통보를 받는 마디에 있다면 그때 그 마디의 목록은 되쓰기가 생길 때까지 기억기의 그 행을 유효상태에서 그대로 유지하도록하는것을 지적하는 시작주소를 기억하고 있는것이 필요하다. 다른 처리장치(국부 혹은 원격)가 무효인 항을 요구하면 그때 국부등록부는 자료를 출력하기전에 비동시쓰기에 의하여 기억기가 갱신되게 하여야 한다.

3. 비균일기억기접근의 우결함

CC-NUMA의 주요한 우점은 특별한 쏘프트웨어의 변화를 요구함이 없이 대칭다중처리장치체계보다는 높은 수준의 병렬화성능을 실현할수 있는것이다. 다중 NUMA 마디들로 임의의 개별적인 마디에 대한 모선통과량은 모선이 조종할수 있는 요구에 제한을 준다. 그런데 만일 대부분의 기억기접근이 원격마디에 대하여 진행된다면 성능은 떨어지기 시작한다. 이 체계성능저하를 피하기 위한 몇가지 방안이 제기되여 있다. 첫째로, 원격인 기억기를 비롯하여 모든 기억기접근을 최소화하기 위하여 L1과 L2캐쉬리용문제가 제기되였다. 만일 대부분의 하드웨어가 좋은 시간국부성을 가지고 있다면 그때 원격기억접근은 크게 문제로 되지 않는다. 둘째로, 쏘프트웨어가 좋은 공간국부성을 가지고 있고 가상기억기가 리용되고 있다면 응용프로그람을 위하여 요구되는 자료는 자주 리용되는 제한된 수의 페지안에 기억되여 있게 되는데 이 페지들은 집행하는 응용프로그람에 들어 가는 극소기억기에 넣을수 있다. 시켄트 설계자들은 그러한 공간국부성이 대표적인

응용프로그람들에서 나타난다는 보고를 제기하였다[Love96]. 마지막으로 가상기억기능은 가상기억페지를 그것을 자주 리용하는 마디에로 옮기는 페지이동기구를 포함시키는 방법으로 강화할수 있다. 실리콘그라픽스설계자들은 이 방법이 가지고 있는 우점을 발표하였다[WHIT97].

CC-NUMA 에는 또한 결함도 있다. [PFIS98]에서는 이에 대한 두가지 문제를 특별히 구체적으로 론의하였다. 첫째로 CC-NUMA 는 명백히 대칭다중처리장치와 같이 볼수 없으며 대칭다중처리장치로부터 CC-NUMA 체계에로 조작체계와 응용프로그람을 이식하기 위하여서는 쏘프트웨어의 변화가 요구된다. 이 변화에는 페지배치와 앞에서 언급된 프로쎄스배치, 조작체계에 의한 부하균형맞추기 등이 포함된다. 둘째로 리용성문제이다. 이것은 복잡한 항목으로서 CC-NUMA 체계의 정확한 실현과 관련된다. 구체적인것은 [PEIS98]에 밝혀 져 있다.

제 6 절. 벡토르계산

일반용대형콤퓨터의 성능이 계속 높아 진다고 할지라도 현재 콤퓨터로 실현할수 없는 응용문제들이 수없이 제기된다. 즉 류체력학, 지질학, 기상학, 원자핵과 플라즈마물리학을 비롯한 학문분야에서 제기되는 수학적문제를 풀기 위하여 콤퓨터가 필요하다 [WILS84].

특히 이 문제들은 높은 정확도를 요구하며 수들의 큰 배렬에 대하여(큰 행렬식에 대하여) 류점수연산을 여러번 반복하여야 하는 특징이 있다. 이 문제의 대부분은 **련속마당모의**라고 하는 범주에 귀착된다. 례컨데 물리적조건은 3 차원안의 면이든가 령역에 의하여 표현될수 있다(례하면 로케트의 표면에 접촉한 공기의 흐름.) 이 면은 격자점을 리용하여 근사화할수 있다. 여러개의 미분방정식은 매개 점에서의 면의 물리적거동을 정의한다. 방정식들은 값과 결수들의 배렬(행렬전개)에 의하여 표현되며 풀이과정은 자료배렬에 대하여 산수연산을 반복하게 된다.

이러한 류형의 문제를 풀기 위하여 슈퍼콤퓨터가 개발되였다. 이 기계는 1s 에 수억회의 류점수연산을 수행한다. 다중프로그람과 I/O 기능을 수행하기 위하여 설계된 대형콤퓨터와는 달리 슈퍼콤퓨터는 앞에서 서술한 수학적계산을 최적화한다.

슈퍼콤퓨터는 가격문제와 시장문제로 하여 리용에서 제한성이 있다. 이 기계들은 연산에 리용되는것은 적고 대부분은 과학쎈터들과 과학 및 공학연구기능을 가진 정부기관들에서 쓰인다. 콤퓨터기술의 다른 령역과 마찬가지로 슈퍼콤퓨터의 성능을 높이는데서 일정한 요구가 제기된다. 류체력학과 핵물리분야의 일부 응용에서 현대 콤퓨터가 이틀이상이 걸려야 하는 10^{13}회의 산수연산이 하나의 문제로 된다[LEVI82]. 그러므로 슈퍼콤퓨터의 기술과 성능을 계속 발전시키고 있다.

벡토르연산을 위한 요구를 실현하기 위하여 설계된 새로운 다른 체계 즉 **배렬처리장치체계**가 있다. 비록 슈퍼콤퓨터가 벡토르계산을 최적화하지만 그것은 스칼라처리과제와 일반자료처리과제를 조종할수 있는 일반용콤퓨터에 불과하다. 배렬처리장치는 스칼라처리를 하지 않으며 그것들을 프로그람의 벡토르화된 부분만을 실행하기 위하여 대형콤퓨터와 소형콤퓨터의 사용자들의 견지에서는 주변장치처럼 동작하도록 구성한다.

1. 벡토르계산방법

슈퍼콤퓨터나 배렬처리장치체계를 설계하는 열쇠는 그 기본과제가 류점수의 배렬 혹은 벡토르산수연산을 수행한다는것을 인식하는것이다. 이것은 일반용콤퓨터에서 배렬의 매개 요소에 대하여 반복연산을 요구한다. 례하면 두수의 벡토르(1차원배렬) A와 B에 대하여 보기로 하자. 방법은 두 원소의 더하기를 하는것으로 된다(그림 16-11). 이 계산을 고속화하기 위하여서는 어떻게 해야 하는가? 그 대답은 명백히 연산의 병렬화이다.

$$\begin{bmatrix} 1.5 \\ 7.1 \\ 6.9 \\ 100.5 \\ 0 \\ 57.9 \end{bmatrix} + \begin{bmatrix} 2.0 \\ 39.7 \\ 1000.003 \\ 11 \\ 21.1 \\ 19.7 \end{bmatrix} = \begin{bmatrix} 3.5 \\ 46.8 \\ 1006.93 \\ 111.5 \\ 21.1 \\ 79.4 \end{bmatrix}$$

그림 16-11. 벡토르더하기

벡토르계산의 병렬화를 실현하기 위하여 여러가지 방법을 취하고 있다.

이것을 실례를 통하여 보여 주었다. 벡토르적(곱하기) C=A×B를 보기로 하자. 여기서 A,B 그리고 C는 N×N행렬이다. C의 매개 원소에 대한 공식은 다음과 같다.

$$c_{i,j} = \sum_{K=1}^{N} a_{i,k} \times b_{k,j}$$

여기서 A, B, C는 각각 원소 $a_{i,j}$, $b_{i,j}$ 그리고 $c_{i,j}$로 되여 있다. 그림 16-12 ㄱ는 이 계산을 위하여 작성한 포트란프로그람을 보여 주었는데 이 프로그람은 보통 스칼라처리장치에서 실행할수 있다.

```
      DO 100 I = 1,N
      DO 100 J = 1,N
      C(I,J)=0.0
      DO 100 K=1,N
      DO 100 K=1,N
      C(I,J)=C(I,J)+
      A(I,K)*B(K,J)
100   CONTINUE
```

ㄱ)

```
      DO 100 I =1,N
      C(I,J)=C(I,J)+
      A(I,K)*B(K,J)(J=1,N)
100   CONTINUE
```

ㄴ)

```
      DO 50 J =1,N
      FORK 100
50    CONTINUE
      J=N
100   DO I =1,N
      C(I,J) =0,0
      DO 200K =1,N
      C(I,J) =C(I,J)+
      A(I,K)*B(K,J)
200   CONTINUE
```

ㄷ)

그림 16-12. 행렬곱하기
ㄱ-스칼라처리과정, ㄴ-벡토르처리과정, ㄷ-병렬처리과정

성능을 개선하기 위한 한가지 방법을 **벡토르처리**라고 한다. 이것은 1차원벡토르자료에 대한 조작이 가능하다는것을 예측할수 있다. 그림 16-12 ㄴ는 벡토르계산을 규정하는 새로운 명령형식으로 된 포트란프로그람이다. 기호(J=1,N)는 주어 진 간격안의 모든 첨수 J에 대한 연산이 한번의 조작에 의하여 수행된다는것을 의미한다. 이것이 어떻게 실현될수 있는가를 간단히 보기로 하자.

그림 16-12 ㄴ의 프로그람은 i번째 렬의 모든 원소들을 병렬로 계산하는것을 보여

주었다. 렬안의 매개 원소는 합계산되며 합계산들은 병렬이라기보다 직렬적으로 진행된다. 스칼라알고리듬에 따르는 N3 스칼라곱하기연산들과 비교해 볼 때 이 N2 벡토르곱하기연산만이 이 알고리듬에 따라 수행된다.

다른 방식인 **병렬처리**과정을 그림 16-12 ㄷ에 보여 주었다. 이것은 병렬로 동작할수 있는 N 개의 독립적인 처리장치로 되여 있는것을 전제로 한다. 처리장치를 효과적으로 리용하기 위하여 여러 처리장치들에 어떻게 해서든지 계산을 나누어 시켜야 한다. 여기서 두개의 내정값이 처리를 시작하게 한다. 한편 원시프로쎄스는 FORK 다음의 명령에서 실행을 계속한다. FORK 의 모든 실행은 새로운 프로쎄스를 낳는다. FORK N 명령문은 N 개의 독립적인 프로쎄스들이 JOIN 다음의 명령에서 실행을 계속하는 프로쎄스에 련결되게 한다. 조작체계는 이 련결을 조종하며 그로 하여 N 개의 모든 프로쎄스들이 JOIN 명령에 도달하였을 때까지는 실행을 계속하지 않는다.

그림 16-12 ㄷ의 프로그람은 벡토르처리프로그람의 거동을 모방하여 작성한것이다. 병렬처리프로그람에서 C 의 매개 행은 개별적인 프로쎄스에 의하여 계산된다. 따라서 주어 진 C 의 렬안의 원소는 병렬로 처리된다.

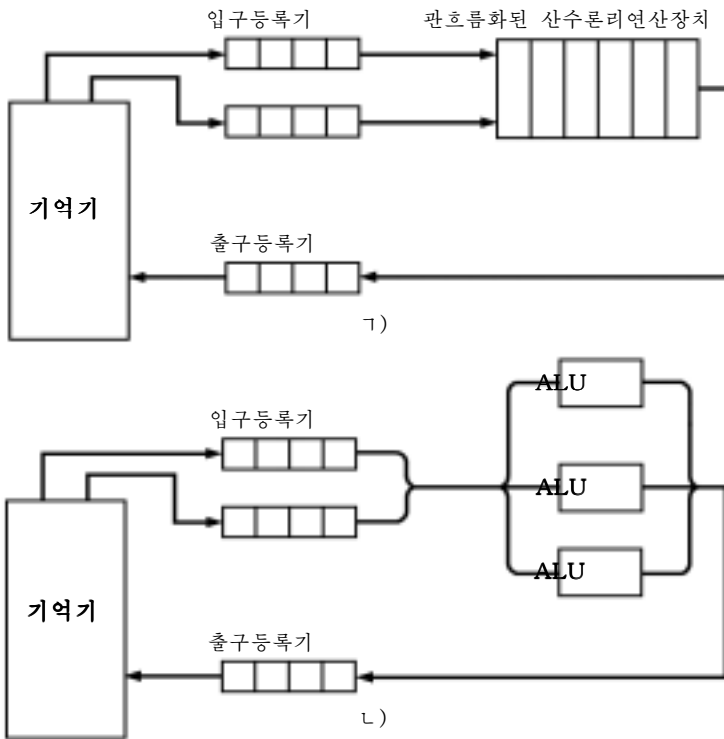



그림 16-13. 벡토르계산에 대한 방식

ㄱ- 관흐름화된 산수론리연산장치, ㄴ- 병렬산수론리연산장치

다음으로 론리적 및 방식적인 견지에서 벡토르계산방법을 고찰한다. 이제 이 방법을 실현하는데 리용할수 있는 처리장치의 류형들을 보기로 하자. 여러가지 류형의 기본방식이 리용되였거나 리용되고 있다. 그가운데서 다음의 세가지 류형이 두드러지게 리용되고 있다.

- 관흐름 ALU
- 병렬 ALU

- 병렬처리장치

그림 16-13 에 세가지 류형가운데서 앞의 두가지를 보여 주었다. 관흐름에 대해서는 제 11 장에서 이미 론의하였다. 여기서는 두 개념을 ALU 의 조작에까지 확장하였다. 류점수연산은 비교적 복잡하기때문에 류점수연산을 여러개 단으로 분해하되 매단에 서로 다른 자료묶음을 병행적으로 조작할수 있도록 한다.

이것을 그림 16-14 ㄱ에 주었다. 류점수더하기는 4 개 즉 비교, 밀기, 더하기와 정규화로 나누어 져 있다(그림 8-22). 수의 벡토르들은 차례로 첫번째 단에 입구된다. 처리순서에 따라 4 개의 서로 다른 수의 묶음은 관흐름안에서 병행적으로 조작되게 된다.

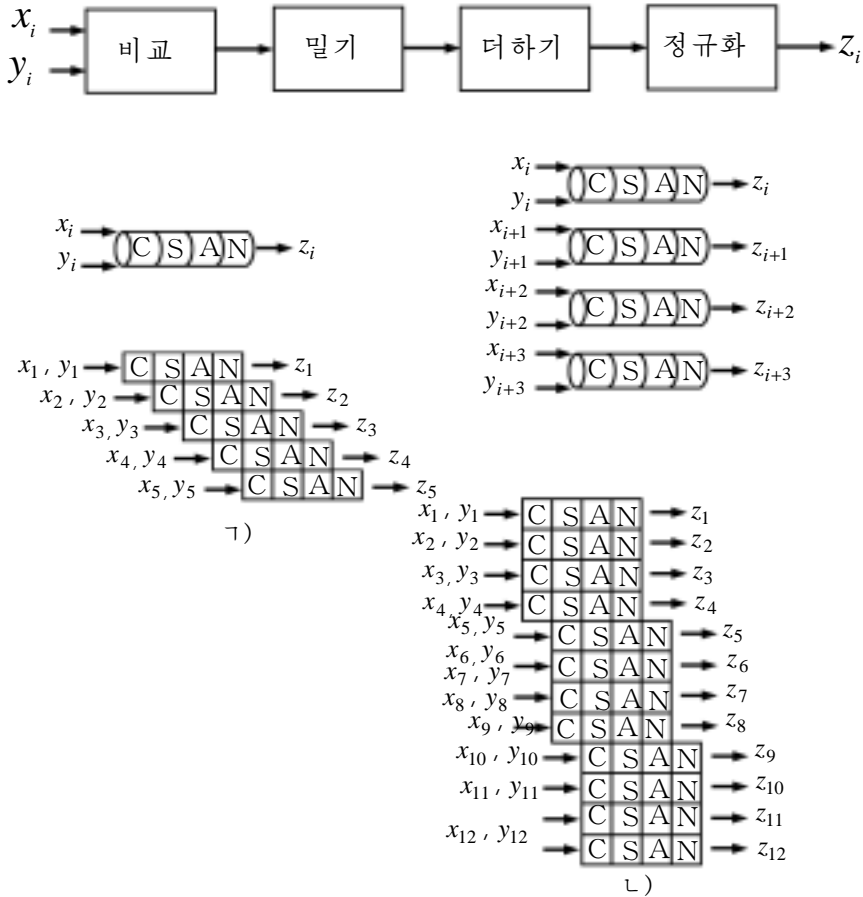



그림 16-14. 관흐름처리

ㄱ-관흐름화된 산수-론리연산장치, ㄴ- 4 개의 병렬산수-론리연산장치

이 구성은 벡토르처리에 적합하다는것이 명백하다. 이것을 알기 위하여 제 11 장에서 서술한 명령관흐름에 대하여 고찰해 보자. 처리장치는 꺼내기명령과 처리명령의 주기를 반복한다. 분기가 없으면 처리장치는 계속하여 다음주소에서 명령꺼내기를 한다. 결과적으로 관흐름이 계속 차 있게 되며 시간이 절약된다. 류사한 방법으로 관흐름산수-론리연산장치는 순차주소로부터 자료의 흐름을 입력시키면 시간을 절약한다. 하나의 분리된 류점수연산에서는 관흐름에 의하여 속도가 높아 지지 않는다. 속도제고는 연산수벡토르가 ALU 에 들어 올 때만 실현된다. 조종장치는 전체 벡토르가 다 처리될 때까지 ALU 를 통하여 자료처리를 반복한다.

벡토르요소가 주기억기로부터가 아니라 등록기들에서 유용하다면 관흐름조작은 더 강화된다. 이것은 실제로 그림 16-13 ㄱ에서 이미 제기하였다. 매개 벡토르연산수의 원소들은 벡토르등록기의 블로크에 넣어 진다. 벡토르등록기는 동일한 등록기들의 묶음에 불과한다. 그 결과는 다시 벡토르등록기에 넣어 진다. 따라서 대부분의 연산은 등록기들을 리용하여 다시넣기, 쓰기조작만을 할수 있으며 벡토르연산의 시작과 끝에서만 기억기접근을 요구한다.

그림 16-14 에 보여 준 기구를 **연산속의 관흐름처리**방식이라고 한다. 즉 벡토르연산수들에 적용되는 하나의 산수연산(례하면 C=A+B)을 가지며 그다음 관흐름처리는 여러개의 벡토르원소(요소)들이 병렬로 처리되게 한다. 이 기구를 **연산의 관흐름처리**방식이라고 한다. 후자의 경우에 산수벡토르연산과정이 있으며 명령관흐름은 처리속도를 높이는데 리용된다. 련쇄화에 대한 기본규칙은 다음과 같다. 벡토르연산은 연산수벡토르의 첫 원소가 유용하며 기능장치(례하면 더하기, 덜기, 곱하기, 나누기)들이 동작가능한 상태(자유로운 상태)에 있으면 즉시에 벡토르연산을 시작한다.

실례로 련쇄는 어느 한 기능장치로부터 출력되는 결과를 곧바로 다음기능장치에 입력시키며 그것이 또 다른 기능장치에 입력되는 식으로 련결한다. 벡토르등록기들을 리용하면 중간결과는 기억기에 기억되지 않으며 그것을 만들어 낸 벡토르연산이 끝날 때까지 리용할수 없다.

례하면 C=(s×A)+B 를 계산할 때 (여기서 A, B, C 는 벡토르이고 s 는 스칼라이다.) 크레이(Cray)슈퍼콤퓨터는 한번에 세개의 명령을 수행한다. 넣기를 하여 꺼낸 원소(요소)들은 즉시에 관흐름곱하기장치에 들어 가며 그 적은 관흐름더하기장치에 보낸다. 그리고 더한 결과는 더하기장치나 더하기연산을 끝내자마자 벡토르등록기에 넣어 진다.

- 벡토르넣기 A →벡토르등록기(VR1)
- 벡토르넣기 BC → VR2
- 벡토르곱하기 s×VR1 → VR3
- 벡토르더하기 VR3+ VR2 → VR4
- 벡토르쓰기 VR4 → C

두번째와 세번째명령은 련쇄화될수 있는데 왜냐하면 그것들은 서로 다른 기억주소와 등록기들을 가지고 있기때문이다. 4 번째 명령은 두번째와 세번째명령의 결과를 요구하지만 이것은 또한 그것들과 련쇄될수 있다. 벡토르등록기 2와 3의 첫 원소들이 유용하게 되면 즉시에 4번째 명령에 대한 조작이 시작된다.

여러개의 산수-론리연산장치를 넣어 하나의 조종장치에 의하여 조종하게 하는것이다. 이 경우에 조종장치는 산수-론리연산장치들이 병렬로 동작하도록 자료전송을 조종한다. 이것은 매개 병렬산수연산장치에 대하여 관흐름화를 실현할수 있게 한다. 이것을 그림 16-14 ㄴ에 4개의 산수-론리연산장치가 병렬로 동작하는 경우로 보여 주었다.

관흐름화된 내부구성을 가짐으로써 병렬산수-론리연산장치의 구성은 벡토르처리에 알맞게 되였다. 조종장치는 모든 원소들이 처리될 때까지 순환식으로 산수-론리연산장치들에 벡토르원소들을 주기 위한 경로를 조종한다. 이러한 류형의 내부구성은 단일산수-론리연산장치의 구성보다 훨씬 복잡하다.

벡토르처리를 위한 또 한가지 방법은 다중병렬처리장치를 리용하는것이다. 이 경우에 과제를 여러 처리장치에 나누어 주어 병렬로 실행할수 있게 하여야 한다. 이 구성방식은 병렬처리장치들을 조종할수 있게 쏘프트웨어와 하드웨어를 잘 리용하기만 하면 그

효과가 대단하다. 그러나 이것은 비록 몇가지 제품이 나오긴 했지만 아직 미개척지이다 [GEHR88].

우리는 제 16 장 제 1 절의 분류를 확장하며 그림 16-15 에서 보여 준것과 같은 이 새로운 구성방식을 론의한다. 콤퓨터의 구성방식은 하나 혹은 그이상의 조종장치의 수에

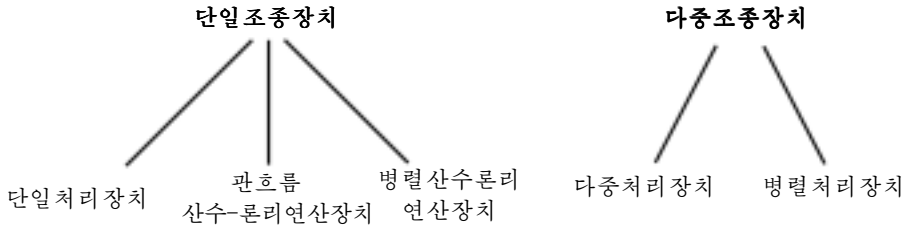



그림 16-15. 콤퓨터구성의 분류법

의하여 구별할수 있다. 여러조종장치라는것은 다중처리장치라는 말을 의미한다. 선행한 고찰에 따라 다중처리장치들은 주어 진 과제에 대하여 협동적으로 동작하므로 **병렬처리장치체계**라고 한다.

독자들은 문헌에서 찾아 보게 되는것과 같은 일부 련관성이 없는 용어들을 알게 될것이다. 비록 병렬 ALU 구성방식이 벡토르처리를 목적으로 설계되고 앞에서 이미 론의한것처럼 병렬처리장치의 구성방식도 또한 벡토르처리를 위하여 설계되였다고 하지만 벡토르처리장치의 구성방식이라고 할 때 그것은 관흐름 ALU 구성방식을 의미하는것이다. 세가지 구성방식가운데서 어느 하나가 배렬처리를 위하여 최적화되여 있다고 해도 **배렬처리**기본방식은 본질에 있어서 병렬 ALU 구성방식이라고 말할수 있다. 배렬처리장치는 일반용처리장치들에 증설되여 보조적으로 벡토르계산을 담당한다. 배렬처리장치방식은 관흐름 ALU 든가 병렬 ALU 방식에 리용할수 있다.

최근에 관흐름 ALU 구성방식의 콤퓨터들이 시장에서 주류를 이루고 있다. 관흐름체계들은 다른 두 방식에 비하여 덜 복잡하다. 그의 조종장치와 조작체계프로그람들은 다원배치(할당)에서의 효과성과 높은 성능을 보장할 목적에서 개발되였다. 이 절의 나머지 부분에서는 이 방식에 대하여 실례를 들어 좀더 구체적으로 설명한다.

2. IBM 3090 벡토르기구

벡토르처리를 위한 관흐름 ALU 조직의 대표적인 실례로 IBM 370 구성방식을 위하여 개발되였으며 고성능 IGBM 3090 계렬에서 실현된 벡토르기구를 들수 있다. 이 기구는 기본체계에 보충하는 식으로 된 선택품이지만 높은 통합성을 가지고 있다(이 기구는 기본체계에 반드시 장비되여 있는것은 아니다.). 필요에 따라 선택하여 확장하는 식으로 되여 있는데 결합이 잘된다.

IBM 기구는 여러개의 벡토르등록기를 리용하게 되여 있다. 매개 등록기는 실제로 여러개의 스칼라등록기들의 묶음으로 되여 있다. 벡토르더하기 C=A+B 를 계산하기 위하여 벡토르 A 와 B 가 두개의 벡토르등록기에 넣어 진다. 다시 이 등록기로부터 출력된 자료들은 가능한 높은 속도로 ALU 를 통과한 다음 (ALU 에 연산된 다음) 그 결과가 세번째 벡토르등록기에 넣어 진다. 계산은 반복계산과 블로크안의 등록기에 입력자료의 넣기조작을 일반 ALU 조작보다 훨씬 더 빠른 속도로 진행하여 결과를 출력한다.

조직

IBM 벡토르기본방식과 그와 류사한 관흐름벡토르 ALU 방식은 스칼라산수명령의 순환이상으로 성능제고를 보장하는데 여기에는 세가지 방법이 있다.

- 벡토르자료구조가 고정되여 있으면서 미리 결정되여 있으므로 순환내부에 있는 명령들을 보다 더 빠른 내부(하드웨어 혹은 마이크로코드) 기계조작(연산)코드로 바꿀수 있게 한다.
- 여러개의 순차적인 벡토르원소(요소)들에 대한 자료접근과 산수연산은 관흐름조작에서와 같이 병행으로 하든가 여러개의 원소(요소)에 대한 연산을 병렬로 수행한다.
- 중간결과를 기억하는데 벡토르등록기를 리용하므로 기억기참조의 증가를 피할수 있다.

그림 16-16에 벡토르기구의 일반적구성을 보여 주었다. 비록 벡토르기구가 물리적으로 분리되여 있으면서 처리장치에 보충적으로 부가되게 되여 있는것으로 보이지만 그 구성방식은 System/370 의 연장이며 이와 호환성을 가진다. 벡토르기구는 다음과 같은 방법으로 System/370 에 통합되여 있다.

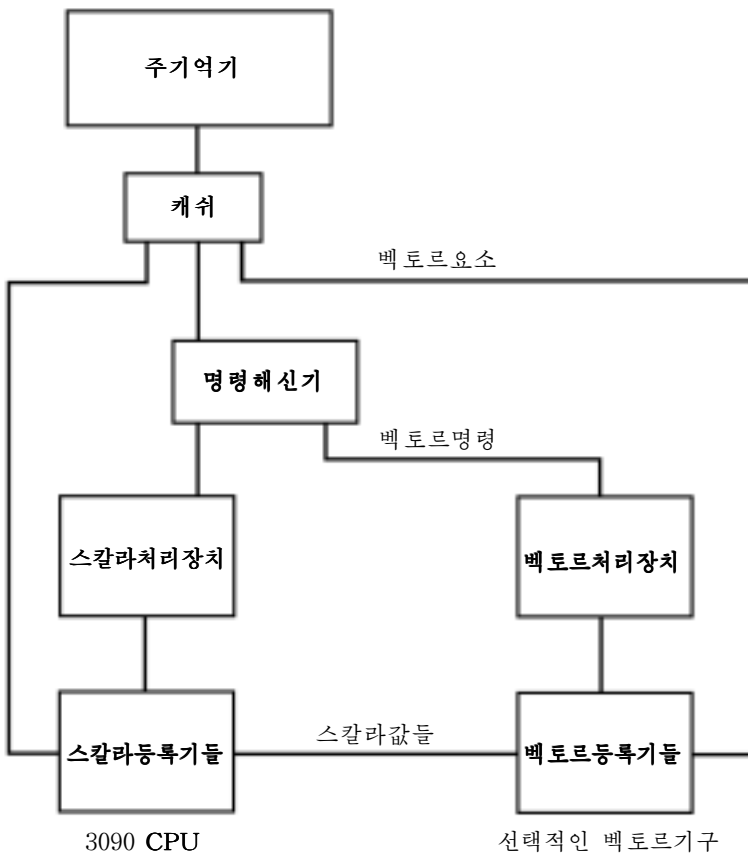



그림 16-16. 벡토르기구를 가진 IBM 3090

- 현존 System/370 명령들은 모든 스칼라연산에 리용된다.
- 개별적벡토르원소에 대한 산수연산은 대응하는 System/370 스칼라명령에 의하여 얻은 결과와 완전히 같다. 례하면 설계에서 한가지 문제가 류점수나누기연산결과에서의 관심사로 되였다. 결과가 정확해야 한다면 스칼라류점수나누기를 해야 하며 혹은 근사계산을 해도 된다면 높은 속도는 실현할수 있지만 때때로 아래자리비트에서 하나 혹은 여러개의 비트가 오유로 될수 있지 않겠는가? 이 결정은 성능에서 손실을 적게 보면서도 System/370 과 완전한 호환성을 보장할수 있게 하였다.
- 벡토르명령들은 새치기할수 있으며 그에 대한 례외처리는 System/370 프로그람새치기방식과 호환성 있게 해당한 동작을 취한후 새치기점에서부터 다시 시작한다.
- 산수연산례외처리들은 System/370 의 스칼라산수명령과 같거나 그의 확장이거나 또는 례외처리로 되여 있으며 류사한 고정루틴(보조프로그람)을 리용할수 있다. 이것을 관리하기 위하여 례외처리에 의하여 영향을 받는(례하면 자리넘침)벡토르등록기안의 주소를 지적하는 벡토르새치기참조부가 리용된다. 따라서 벡토르명령의 례외처리가 다시 시작될 때 벡토르등록기안의 해당한 위치가 접근된다.
- 벡토르자료는 표준방법으로 조종되는 페지기정값을 가진 가상기억기안에 존재한다.

이러한 준위에서 통합은 일련의 우점을 가지고 있다. 현존조작체계들은 크게 확장하지 않고 벡토르기구를 지원할수 있다. 현존응용프로그람들, 언어콤파일러들과 다른 쏘프트웨어들도 갱신하지 않고 그대로 실행할수 있다. 벡토르기구를 리용할수 있는 쏘프트웨어를 희망에 따라 변경할수 있다.

등록기

벡토르기구의 설계에서 기본항목은 연산수가 등록기안에 있는가 기억기안에 있는가 하는것이다. IBM 내부구성을 **등록기-등록기방식**이라고 하는데 그 리유는 벡토르연산수에 대한 입력과 출력이 벡토르등록기들안에서 단계별로 진행되기때문이다. 이 방법은 크레이슈퍼콤퓨터에서 또한 실현되였다. 콘트롤데이터콤퓨터에서 리용된 방식은 기억기로부터 연산수를 직접 얻게 되여 있다. 벡토르등록기리용의 우점은 프로그람작성자라든가 콤파일러작성자가 그것들을 마음대로 쓸수 있다는것이다. 례하면 벡토르등록기의 길이가 K 이고 처리되는 벡토르의 길이가 N>K 이라고 하자. 이 경우 벡토르순환이 진행되여야 하는데 여기에서 한번에 K 개의 원소에 대한 조작이 진행되며 N/K 번 순환이 반복된다. 벡토르등록기방식의 주요한 우점은 연산속도가 뜬 주기억기에서 진행되지 않고 대신 주로 등록기들과 진행하는것이다.

등록기를 리용하여 실현할수 있는 속도제고를 그림 16-17 에서 보여 주었다. 그림에 있는 포트란프로그람은 벡토르 A 와 벡토르 B 를 곱하여 벡토르 C 를 얻은것으로서 여기에서 매개 벡토르는 실수부(AR, BR, CR)와 허수부(AI, BI, CI)를 가진다. 3090 콤퓨터는 하나의 처리장치가 박자 혹은 주기(읽기, 쓰기)마다 주기억접근을 할수 있으며 주기마다 읽기인 경우는 두번 접근, 쓰기인 경우는 한번 접근을 할수 있는 등록기들을 가지는것과 함께 론리연산장치에서는 한주기에 하나의 결과를 만들수 있다. 두개의 원천연산수와 하나의

결과를 규정하는 명령을 리용하는 경우를 보기로 하자 [4]. 그림의 ㄱ는 기억기-기억기명령을 리용하는 경우 매개 계산의 반복에 대하여 18 주기가 요구된다는것을 보여 주고 있다. 순수한 등록기-등록기조작을 진행하면 벡토르량은 계산에 앞서 벡토르등록기에 넣어 지게 되며 그후에 기억기에 기억된다. 벡토르가 큰 경우에 대하여 그 고정된 위반은 상대적으로 작다. 그림 16-17 ㄷ는 하나의 명령안에서 기억기연산수와 등록기연산수를 규정하는 시간이 한번의 반복에 대하여 10 주기까지 시간을 줄일

포트란루틴

```
         DO100J=1. 50
         CR (J)=AR (J) *BR(J) -AI(J) *BI (J)
100      CI (J)=AR (J) *BI (J) -AI(J) *BI (J)
```

조작	주기
AR (J) *BR (J) → T1 (J)	3
AI (J) *BI (J) → T2 (J)	3
T1 (J) −T2 (J) → CR (J)	3
AR (J) *BI (J) → T3 (J)	3
AI (J) *BR (J) → T4 (J)	3
T3 (J) +T4 (J) → CI (J)	3
TOTAL	18

ㄱ)

조작	주기
AR (J) → V1 (J)	3
BR (J) → V2 (J)	3
VI (J) *V2(J) → V3 (J)	3
AI (J) → V4 (J)	3
B1 (J) → V5 (J)	3
AR (J) *V5(J) → V6 (J)	3
AR (J) → V7 (J)	3
BR (J) → CR (J)	3
VI (J) *V2(J) → V8 (J)	3
AI (J) → V9 (J)	3
B1 (J) → V0 (J)	3
AR (J) *V5(J) → C1 (J)	3
TOTAL	18

ㄴ)

조작	주기
AR (J) → V1 (J)	3
V1 (J) *BR (J) → V2 (J)	3
AI (J) → V3 (J)	3
V3 (J) *BI (J) → V4 (J)	3
V2 (J) −V4 (J) → V5 (J)	3
V5 (J) +T4 (J) → CR (J)	3
V1 (J) *BR (J) → V6 (J)	3
V3 (J) −T2 (J) → V7 (J)	3
V6 (J) *BI (J) → V8 (J)	3
V8 (J) *BR (J) → T4 (J)	3
TOTAL	18

ㄷ)

조작	주기
AR(J) → V1(J)	1
V1(J)*BR(J) → V2(J)	1
AI (J) → V3(J)	1
V2(J)−V3 (J) *BI(J) → V2(J)	1
V2(J) → CR(J)	1
V1(J)*BI (J) → V4(J)	1
V4(J)+V3 (J) *BR(J) → C4(J)	10
V4(J) CI (J)	1
TOTAL	8

ㄹ)

Vi=벡토르등록기
AR, BR, AI, BI=기억기안의 연산수
Ti=기억기의 림시연산수

그림 16-17. 벡토르계산을 위한 프로그람들
ㄱ-기억기-기억기, ㄴ-등록기-등록기, ㄷ-기억기-등록기, ㄹ-합성명령

수 있다는것을 보여 주고 있다. 이 명령의 후자의 류형은 벡토르구성방식안에 넣어

[4] 370/390 방식에 대하여 다만 세개의 연산명령(등록기와 기억기명령, RS)만이 두개의 등록기안에 두개의 연산수와 기억기안에 한개의 연산수를 규정한다. 실례의 한 부분을 통하여 주기억기안에 있는 모든 연산수들에 세개의 연산명령이 있다는것을 알수 있다. 이것은 비교의 목적으로 진행되였으며 실제로 그러한 명령형식은 벡토르방식을 위하여 만들어 놓았다.

져 있다.[5]

그림 16-18 은 IBM3090 벡토르기구의 한 부분인 등록기들을 보여 준다. 여기에는 16 개의 32bit 등록기가 있다. 벡토르등록기는 또한 호상 결합되여 8 개의 64bit 등록기로 될수도 있다. 임의의 등록기들은 옹근수값이든가 류점수값을 취할수 있다. 따라서 벡토르등록기들은 32bit 와 64bit 옹근수값과 32bit 와 64bit 의 류점수값을 보관하는데 리용할수 있다.

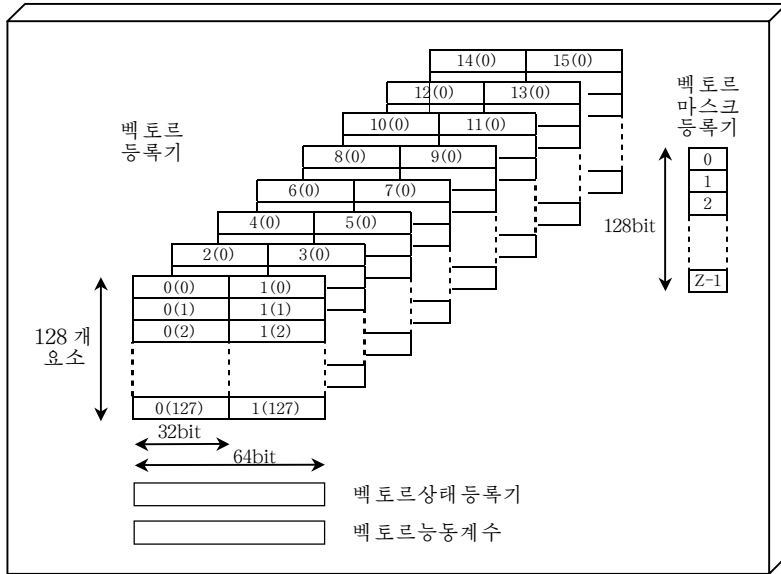



그림 16-18. IBM3090 벡토르기구의 등록기들

이 구성방식은 매개 등록기가 8~512 개의 스칼라요소들을 기억할수 있다는것을 규정하고 있다. 실지길이의 선택에서는 설계에서 동시에 실현할수 없는 두가지 문제가 있다. 벡토르연산을 수행하는 시간은 구체적으로는 관흐름시작(동작의 시작)과 등록기를 채우기 위한 내부처리시간과 벡토르원소에 대한 한주기시간으로 구성되여 있다. 따라서 많은 수의 등록기요소들을 리용하면 계산에 필요한 상대적시작시간을 줄일수 있다. 그런데 이 효과성은 프로쎄스의 절환에 따르는 벡토르등록기들의 보관과 복귀를 위하여 요구되는 보충적인 시간과 실제적인 가격과 공간의 제한성과의 관계에서 균형을 보장하여야 한다. 이러한 고찰에 기초하여 현행 3090 콤퓨터를 실현할 때 등록기당 128 개 요소를 가지게 설계되였다.

벡토르기구를 구성함에 있어서 세개의 등록기를 더 보충하는것이 필요하다. 벡토르마스크등록기는 벡토르등록기안의 어느 요소들을 특별한 조작을 하는 경우에 처리하여야 하는가 하는것을 선택하는데 리용되는 마스크비트들로 구성되여 있다. 벡토르상태등록기는 벡토르계수와 같은 조종마당으로 되여 있는데 이것은 벡토르등록기안의 얼마만한 요소들이 처리되였는가 하는것을 결정한다. 벡토르동작계수(등록기)는 벡토르명령을 실행하는데 소비된 시간을 보관한다.

합성명령

앞에서 론의한것처럼 명령실행은 성능을 개선하기 위하여 련쇄방식을 리용하여 중

---

[5] 후에 론의되는 복합명령문은 더 감소되지 않는다.

첩시켜야 한다. IBM/370 의 벡토르기구설계자들은 여러가지 리유로부터 이 가능성을 고려하지 않았다. System/370 구성방식(가상기억기관리에 의한 효과를 비롯하여)은 복잡한 새치기를 조종하기 위하여 확장하지 않으면 안되였으며 이에 대응하여 쏘프트웨어의 변화가 요구되였다. 보다 기본으로 되는 문제는 발생된 련쇄에 따르는 벡토르기구안에서 조종장치와 등록기접근경로의 보충으로 인한 가격문제이다.

그대신 세가지 조작이 제공되고 있는데 그것은 벡토르계산에서 제일 많이 쓰이는 과정인 더하기, 덜기에 의한 곱하기연산이든가 혹은 합계산을 하나의 명령(조작코드)으로 결합하는것이다. 례하면 기억기-등록기 MULTIPLY-AND-ADD 명령은 기억기로부터 벡토르를 꺼내여 그것을 등록기에 있는 벡토르와 곱한 다음 그 적을 등록기안의 세번째 벡토르에 더한다. 그림 16-17 의 실례에서 합성명령 MULTIPLY-AND-ADD 와 MULTIPLY-AND-SUBTRACT 를 리용함으로써 반복하는 전체 시간은 10 으로부터 8 로 줄어 들게 된다.

련쇄과정과는 달리 합성명령은 중간결과를 림시기억하기 위하여 등록기를 더 보충할 필요는 없으며 이것들은 등록기접근보다 하나 적게 요구한다. 례를 들어 다음과 같은 련쇄과정을 보기로 하자.

$$A \rightarrow VR1$$
$$VR1 + VR2 \rightarrow VR1$$

이 경우 벡토르등록기 VR1 에 두번의 기억이 요구된다. IBM 구성방식에는 기억기-등록기 ADD 명령이 있다. 이 명령에 있어서 합은 오직 VR1 에 넣어 진다. 합성명령은 또한 여러개의 명령의 동시집합을 기계상태로 서술하는데서 제기되는 요구들을 피할수 있으며 그로하여 조작체계와 새치기조종에 의하여 생기는 보장과 복귀의 상태를 간소화할수 있다.

명령모임

표 16-13 은 벡토르방식을 위하여 정의된 산수-론리연산명령을 종합한것이다. 여기에 기억기-등록기넣기와 등록기-기억기쓰기명령이 더 보충되여 있다. 참고할것은 대부분의 명령들이 세개의 연산수형식을 취한다는것이다. 또한 일부 명령들은 연산수의 주소에 따라 여러가지 종류가 있다. 원천연산수는 벡토르등록기(V), 기억기(S)든가 혹은 스칼라등록기(Q)가운데서 어느 하나로 된다. 목적지(결과를 넣는곳)는 비교명령을 제외하고는 항상 벡토르등록기이며 결과는 벡토르마스크등록기에 들어 간다. 이 모든 종류를 고려하면 전체 조작코드의 수는 171 개이다. 일단 기계가 산수연산장치와 자료통로를 기억기, 스칼라등록기, 벡토르등록기로부터 벡토르관흐름에로 연산수를 전송하는데 리용된다면 주요한 하드웨어가격에서 문제가 제기된다. 그 구성방식은 가격에서 별반 차이 없이 이 등록기들과 관흐름들의 리용에 대한 풍부한 명령의 종류를 제공할수 있다.

표 16-3 에 있는 대부분의 명령들은 명백하다. 두개의 합계산명령은 설명을 더 명백히 해준다. 축적연산은 하나의 백토르요소(ACCUMULATE)라든가 두개 벡토르의 적에 대한 요소(MLLTIPLY-AND-ACCUMULATE)와 함께 더한다. 이 명령들은 흥미 있는 설계문제를 제기한다. ALU 관흐름의 우점을 충분히 발휘하여 이 연산이 가능한 빨리 실행되게 하는것이 필요하다. 여기서 어려운 문제는 관흐름안에 있는 두수의 합을 여러 주기동안 리용할수 없는것이다. 따라서 벡토르의 세번째 요소는 그 두 요소가 전체 관흐름을 통하여 출력될 때까지 첫 두 요소의 합을 더할수 없는것이다. 이 문제를 극복하기 위하여 벡토르요소들은 4 개의 부분합을 만드는 방법으로 더하게 된다. 특히 요소 0, 4, 8, 12,···,124 는 부분합 0 을 만들기 위하여, 요소 1, 5, 9, 13, ···,125 는 부분합 1 을 만

들기 위하여 그리고 요소 3, 6, 11, 15, •••,127은 부분합을 만들기 위하여 더해 진다. 이 개개의 부분합은 높은 속도로 관흐름을 통하여 진행할수 있는데 그것은 관흐름안에서의 지연은 대체로 4 주기정도이기때문이다. 개별적인 벡토르등록기들은 부분합을 취하는데 리용된다. 원천벡토르의 모든 요소가 처리되였을 때 4 개의 부분합은 함께 더해 져 최종 결과를 얻는다. 이 두번째 과정의 수행은 문제로 되지 않는데 그것은 오직 4개의 벡토르 요소만이 포함되여 있기때문이다.

표 16-3. IBM3090 벡토르성능: 산수 및 론리연산명령

조작	자료형			연산수위치			
	류점수						
	긴형	짧은형	2진 혹은 론리형				
더하기	FL	FS	BI	$V + V \to V$	$V + S \to V$	$Q + V \to V$	$Q + S \to V$
덜기	FL	FS	BI	$V - V \to V$	$V - S \to V$	$Q - V \to V$	$Q - S \to V$
곱하기	FL	FS	BI	$V \times V \to V$	$V \times S \to V$	$Q \times V \to V$	$Q \times S \to V$
나누기	FL	FS	—	$V/V \to V$	$V/S \to V$	$Q/V \to V$	$Q/S \to V$
비교	FL	FS	BI	$V \cdot V \to V$	$V \cdot S \to V$	$Q \cdot V \to V$	$Q \cdot S \to V$
곱한 다음 더하기	FL	FS	—		$V + V \times S \to V$	$V + Q \times V \to V$	$V + Q \times S \to V$
곱한 다음 덜기	FL	FS	—		$V - V \times S \to V$	$V - Q \times V \to V$	$V - Q \times S \to V$
곱한 다음 축적	FL	FS	—	$P + \cdot V \to V$	$P + \cdot S \to V$		
보수	FL	FS	BI	$- V \to V$			
정의 절대값	FL	FS	BI	$\lvert V \rvert \to V$			
부의 절대값	FL	FS	BI	$- \lvert V \rvert \to V$			
최대	FL	FS	—			$Q \cdot V \to Q$	
최대절대값	FL	FS	—			$Q \cdot V \to Q$	
최소	FL	FS	—			$Q \cdot V \to Q$	
론리왼쪽밀기	—	—	LO	$\cdot V \to V$			
론리오른쪽밀기	—	—	LO	$\cdot V \to V$			
론리곱하기	—	—	LO	$V \& V \to V$	$V \& S \to V$	$Q \& V \to V$	$Q \& S \to V$
론리더하기	—	—	LO	$V \mid V \to V$	$V \mid S \to V$	$Q \mid V \to V$	$Q \mid S \to V$
안맞음론리더하기	—	—	LO	$V \oplus V \to V$	$V \oplus S \to V$	$Q \oplus V \to V$	$Q \oplus S \to V$

해설: 자료형

FL 긴 류점수
FS 짧은 류점수
BI 2진옹근수
LO 론리값

연산수위치

V 벡토르등록기
S 기억장치
Q 스칼라(일반 혹은 류점수등록기)
P 벡토르등록기에서 부분합
· 특수연산

참 고 문 헌

[CATA94]는 다중처리장치의 원리를 개괄하고 SPARC 에 기초한 SMP 를 상세히 고찰하고 있다. [STON93]과 [HWAN93] 역시 SMP 를 일부 상세히 서술하고 있다. [PFIS98]은 클라스터에 흥미가 있는 독자들에게 좋은 참고문헌으로 된다. 이 책은 우선 하드웨어와 쏘프트웨어문제점들을 론의하고 SMP 와 NUMA 로 된 클라스터들을 비교하고 있다. 또한 이 책은 SMP 와 NUMA 설계문제점들을 기술적으로 설명하고 있다.

다중처리장치에서 캐쉬일치성과 관련한 문제점들을 가장 훌륭하게 개괄한 책은 [LILJ93]이다. [TOMA93] 역시 캐쉬일치성에 대한 기본론문들을 많이 담고 있다.

벡토르계산에 대해서는 [STON93]과 [HWAN93]을 참고하면 잘 알수 있다.

CATA94 Catanzaro, B. *Multiprocessor System Architectures.* Mountain View, CA: Sunsoft Press, 1994.

HWAN93 Hwang, K. *Advanced Computer Architecture,* New York: McGraw-Hill, 1993.

LILJ93 Lilja, D. “Cache Coherence in Large-Scale Shared-Memory Multiprocessors: Issues and Comparisons.” *ACM Computing Surveys, September 1993.*

PFIS98 Pfister, G.*In Search of Clusters.* Upper Saddle River, NJ: Prentice Hall, 1998.

STON93 Stone, H. *High-Performance Computer Architecture.* Reading, MA: Addison-Wesley, 1993.

TOMA93 Tomasevic, M., and Milutinovic, V. *The Cache Coherence Problem in Shared-Memory Multiprocessors: Hardware Solution.* Los Alamitos, CA: IEEE Computer Society Press, 1993

련습문제

1. α는 n 개의 처리장치에 의하여 동시에 실행할수 있는 프로그람의 퍼센트라고 하자. 나머지코드가 단일처리장치에 의하여 순차적으로 실행되여야 한다고 하자. 매개 처리장치가 x MIPS 의 실행속도를 가진다.
   ㄱ. α, n 그리고 x 의 견지에서 이 프로그람의 독점적실행을 위한 체계를 리용할 때 유효한 MIPS 속도에 대한 식을 유도하시오.
   ㄴ. n=16 이고 x=4MIPS 의 체계전송을 실현할수 있는 α의 값을 결정하시오.
2. 8 개의 처리장치로 된 다중처리장치체계는 20 개의 접촉식테프구동기를 가지고 있다. 실행을 완성하기 위하여 각각 4 개의 테프구동기를 요구하는 체계에 모든 수의 일감이 제기된다. 매개 일감은 그 조작의 완료를 위하여 짧은 기간에 4 번째 구동기를 요구하기전에 오랜 세개의 테프구동기만으로 실행을 시작한다고 하자. 또한 그러한 일감이 무한히 들어 온다고 하자.
   ㄱ. 조작체계의 일정작성기가 4 개의 테프구동기가 유용하지 않으면 일감을 시작하지 않는다고 하자. 일감이 시작되면 4 개의 구동기는 즉시에 할당되며 일감이 끝날 때까지 놓지 않는다. 한번에 진행할수 있는 일감의 최대수는 얼마인가? 이 방식의 이 결과에 의하여 남아 있는 테프구동기의 최대수와 최소수는 얼마인가?
   ㄴ. 테프구동기의 리용률을 개선하며 같은 시간에 체계의 봉쇄(막힘)를 피하기 위하여 새로운 방식을 제기하시오. 한번에 진행할수 있는 일감의 최대수는 얼마인가? 작업하지 않는 테프구동기수의 한계는 얼마인가?
3. 모선에 기초한 다중처리장치에 대한 한번쓰기캐쉬방식에서 어떤 문제를 주목할

수 있는가? 그렇다면 그의 풀이를 제기하시오.

4. 두 대칭다중처리장치구성에서 두처리장치가 주기억기로부터 자료의 같은 행에 대한 접근을 요구하는 조건(환경)을 고찰하자. 두 처리장치는 하나의 캐쉬를 가지고 MESI 규약을 리용한다고 하자. 처음에 두 캐쉬는 그 행에 대한 무효복사를 한다. 그림 16-19 에 처리장치 P1 에 의하여 행 X 를 읽는 과정을 보여 주었다.

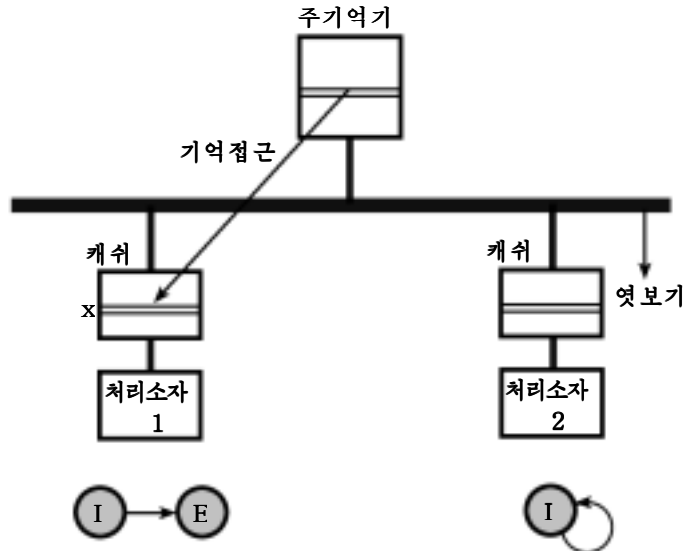



그림 16-19. 처리소자의 행 x 를 읽기

이것이 접근과정에 대한 시작이라면 다음의 과정을 순서대로 그리시오.

1. P2 는 X 를 읽는다.
2. P2 는 X 에 쓴다(P1 에 캐쉬안의 행을 표시).
3. P1 은 X 에 쓴다(P1 에 캐쉬 X 안의 행을 표시).
4. P2 는 X 를 읽는다.

5. 그림 16-20 에 가능한 캐쉬일치성규약을 두개의 상태도로 보여 주었다. 매개 규약을 추리하고 설명한 다음 MESI 와 각각 비교하시오.

6. MESI 규약을 리용하는 L1 과 L2 캐쉬를 가진 대칭다중처리장치에 대하여 보기로 하자. 제 16 장 제 3 절에서 설명한것처럼 4 개 상태가운데서 하나는 L2 캐쉬안의 매개 행과 관련되여 있다. L1 캐쉬의 매개 행에 대하여 4 개 상태가 다 필요되는가? 그렇다면 왜 그런가? 그렇지 않다면 상태나 상태들을 소거해도 될수 있다는것을 설명하시오.

7. 표 16-1 은 IBM S/390 에 대한 3 준위 캐쉬배렬에 의하여 얻어 지는 성능을 보여 준다. 이 문제의 목적은 세번째 준위의 캐쉬를 포함시키는것이 가치가 있겠는가를 결정하는것이다. L1 캐쉬만을 가지고 있는 체계에 대한 접근(처리장치주기의 평균수)를 결정하고 그 값을 1.0 까지 정규화하시오. 그다음 L1 과 L2 캐쉬를 둘다 리용했을 때 정규화된 접근위반과 세 캐쉬를 다 리용했을 때의 접근위반을 결정하시오. 매개 캐쉬에서 개선된 량을 참고로 하여 L3 캐쉬값에 대한 독자의 견해를 말하시오.

8. 다음의 코드토막은 벡토르산수연산식

$$D(I) = A(I) + B(I) \times C(I) \quad (0 \leq I \leq 63 \text{ 인 경우})$$

을 평가하기 위하여 64 번의 실행이 필요하다.

```
Load   R1,   B(I)          /R1 ← Memory (α + I)/
```

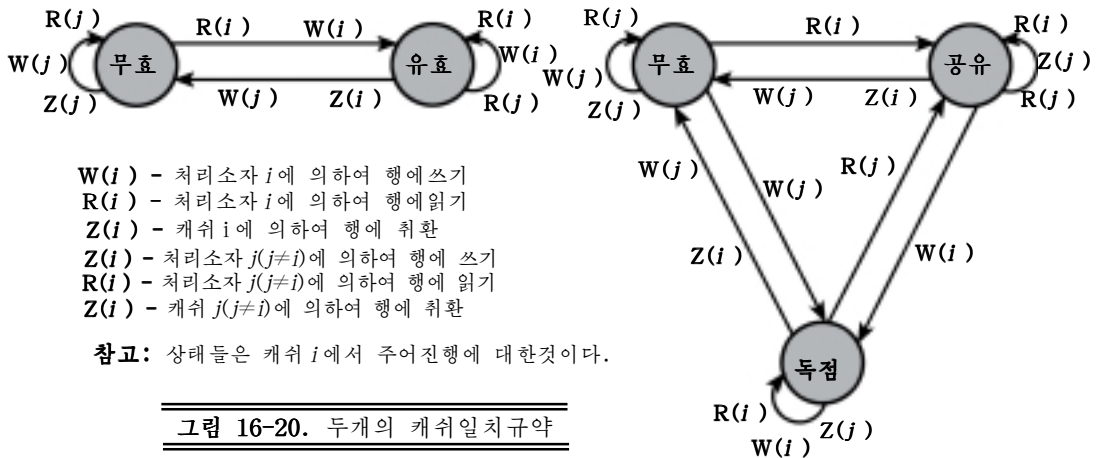
```
Load     R2,   C(I)        /R1 ← Memory (β+I)/
Multiply       R1, R2      /R1 ← (R1)× (R2)/
Load     R3,   A(I)        /R1 ← Memory (γ+I)/
Add      R3,   R1          /R1 ← (R3)× (R1)/
Load     D1,   R3          /Memory θ ← (R3)/
```

여기서 R1, R2 와 R3 은 처리장치안의 등록기들이며 $\alpha, \beta, \gamma, \theta$ 는 배렬 B(I), C(I), A(I) 및 D(I)의 개개에 대한 주기억기안의 시작주소이다. 매개 읽기, 쓰기에 4 박자주기, 더하기에 두박자주기, SIMD 기계안에서 단일처리장치 혹은 하나의 처리장치에서 곱하기를 위하여 8 박자주기라고 하자.

ㄱ. SISD 단일처리장치콤퓨터에서 이 토막을 순차적으로 그리고 반복하여 64 번 실행하는데 요구되는 처리장치의 전체 박자주기수를 계산하시오. 이때 다른 시간지연을 무시한다.

ㄴ. 64 개의 벡토르자료에 대하여 6 개의 동기화된 벡토르명령안의 벡토르연산을 실행하기 위하여 64 개의 처리요소를 가진 SIMD 콤퓨터의 리용을 설명하시오. 그리고 SIMD 기계의 전체 실행기간을 계산하시오. 명령알림과 다른 지연은 무시한다.

ㄷ. SISD 콤퓨터에 비하여 SIMD 콤퓨터의 속도리득은 얼마인가?

그림 16-20. 두개의 캐쉬일치규약

9. 다음의 프로그람을 벡토르화하시오.

10. 단일처리장치는 스칼라방식과 벡토르방식에서 동작할수 있다. 계산속도는 벡토르방식에서 10 배 더 빠르다. 성능검사프로그람을 가지고 이 콤퓨터에서 시간 T를 측정하였다. 이 시간에서 25%는 벡토르방식이고 나머지는 스칼라방식이다.

ㄱ. 우에서 언급된 조건에서 벡토르방식을 리용하지 않는 경우와 비교하여 유효속도제고비를 계산하시오. 또한 앞의 프로그람에서(벡토르방식을 리용하기 위하여 번역된) 벡토르화된 코드의 퍼센트인 α를 구하시오.

ㄴ. 하드웨어를 개량함에 따라 벡토르방식과 스칼라방식사이의 속도비가 배로 된다는것을 설명하시오. 얻어 진 유효속도제고비를 계산하시오.

ㄷ. ㄴ에서 얻어 진 속도제고비가 하드웨어를 개량하는것보다 콤파일러를 개량하여 얻을수 있다는것을 설명하시오. 같은 성능검사프로그람에 비하여 콤파일러에 의하여 지원 받을수 있는 벡토르화비 α는 얼마인가?

부록 Ⅰ. 수자론리

수자형콤퓨터의 연산은 2 진자료에 대한 기억과 처리에 기초하고 있다. 이 책에서는 처음부터 두개의 안정상태가운데서 한 상태에 놓일수 있는 기억요소와 콤퓨터의 각이한 기능을 실현하기 위한 조종신호의 조종밑에 2 진자료에 대한 연산을 진행할수 있는 회로가 있다고 보고 설명을 해왔다. 이 부록에서는 이러한 기억요소와 회로가 조합회로와 순서회로와 같은 수자론리로 어떻게 실현되는가에 대하여 본다. 부록에서는 먼저 불대수를 개괄적으로 고찰하는데 이것은 수자론리의 수학적기초로 된다. 다음으로 문회로의 개념을 고찰하고 마지막으로 문회로로 구성되는 조합회로와 순서회로에 대하여 해설한다.

제 1 절. 불대수

수자형콤퓨터 및 다른 수자체계에서 쓰이는 수자회로는 불대수로 알려 진 수학리론에 의하여 설계되며 그 동작이 해석된다. 이 이름은 영국의 수학자 죠지 불(George Boole)에 대한 경의의 표시로 단것이다. 그는 이 대수학의 기본원리를 1854 년에 그의 론문 **론리 및 확률의 수학적리론을 창시하기 위한 사유의 법칙에 대한 연구**에서 제안하였다. 1938 년에 MIT 에서 전기공학과 연구조수로 있던 클라우드 샤논(Claude Shannon)은 계전기절환회로의 설계문제를 해결하는데 이 불대수를 리용할수 있다는것을 제기하였다[SHAN38]. 샤논의 수법은 그후 수자전자회로의 설계와 해석에 리용되였다. 불대수는 두 분야에서 편리한 도구로 되였다.

- **분석:** 수자회로의 기능을 서술하는 경제적인 방도이다.
- **설계:** 필요한 기능이 주어 진 경우 그 기능을 간단하게 실현하는데 불대수를 적용할수 있다.

임의의 대수와 마찬가지로 불대수도 변수를 리용하여 연산을 한다. 불대수인 경우 변수와 연산은 론리변수와 론리연산으로 된다. 변수는 1(참) 또는 0(거짓)이라는 값을 취할수 있다. 기본론리연산에는 론리곱하기(AND), 론리더하기(OR) 및 론리부정(NOT)이 있는데 이것들은 각각 점, +기호 및 웃막대기와 같은 기호로 표시된다.

$$\text{A AND B} = \text{A} \bullet \text{B}$$
$$\text{A OR B} = \text{A} + \text{B}$$
$$\text{NOT A} = \overline{\text{A}}$$

AND 연산은 량쪽 연산수가 다 참일 때에만 참(2 진값 1)을 얻게 된다. OR 연산은 참가하는 연산수가운데서 어느 하나 또는 량쪽이 참일 때 참이라는 결과를 준다. NOT 연산은 그의 연산수값을 반전시킨다. 실례로 다음식을 고찰해 보자.

$$\text{D} = \text{A} + (\overline{\text{B}} + \text{C})$$

A 가 1 이거나 혹은 B=0 이고 C=1 이면 D 는 1 이 된다. 그렇지 않으면 D 는 0 이다.

표 I-1. 불연산자

P	Q	NOT P	P AND Q	P OR Q	P XOR Q	P NAND Q	P NOR Q
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	1	0
1	1	0	1	1	0	0	0

표기법에 대하여 몇가지 점을 설명할 필요가 있다. 괄호가 없는 경우에는 AND 연산이 OR 연산보다 우선적이다. 또한 모호한것이 없으면 점연산자대신에 변수들을 그냥 련결하여 표시할수 있다. 즉

$$A + B \bullet C = A + (B \bullet C) = A + BC$$

우의 식들은 모두 B 와 C 의 AND 를 진행하고 그다음 그 결과와 A 의 OR 를 취한다는것을 의미한다.

표 I-1 은 **진리값표**로 알려 진 형식으로 기본론리연산들을 정의하고 있다.이 진리값표는 단순히 연산수값의 모든 가능한 조합에 대하여 그 연산값을 목록화한것이다. 표에서는 다른 3 가지 쓸모 있는 연산자인 안맞음론리더하기(XOR), 론리곱하기부정(NAND) 및 론리더하기부정(NOR)에 대해서도 목록화하였다. 두 론리연산수의 XOR 는 두 연산수가운데서 어느 하나만 값 1 을 가질 때 1 이 된다. NAND 기능은 AND 기능의 부정(NOT)이고 NOR 는 OR 의 부정이다.

$$A\ \text{NAND}\ B = \text{NOT}(A\ \text{AND}\ B) = \overline{AB}$$
$$A\ \text{NOR}\ B = \text{NOT}(A\ \text{OR}\ B) = \overline{A + B}$$

이 3 가지 새로운 연산은 어떤 수자회로를 실현하는데 효과적으로 쓸수 있다.

표 I-2 는 불대수의 기본항등식들을 정리한것이다. 식들은 AND 와 OR 연산의 상보적 및 쌍대적성질을 보여 주기 위하여 두줄로 배렬하였다. 두 종류의 항등식들이 있는데 하나는 증명없이 식으로 표시된 기본규칙(또는 **기본원리**)이고 다른 하나는 기본원리로부터 유도될수 있는 항등식이다.

표 I-2. 불대수의 기본항등식

기본원리		
$A \bullet B = B \bullet A$	$A + B = B + A$	교환법칙
$A \bullet (B + C) = (A \bullet B) + (A \bullet C)$	$A + (B \bullet C) = (A + B) \bullet (A + C)$	분배법칙
$1 \bullet A = A$	$0 + A = A$	동일요소
$A \bullet \overline{A} = 0$	$A + \overline{A} = 1$	반전요소
다른 항등식		
$0 \bullet A = 0$	$1 + A = 1$	
$A \bullet A = A$	$A + A = A$	
$A \bullet (B + C) = (A \bullet B) \bullet C$	$A + (B + C) = (A + B) + C$	결합법칙
$\overline{A \bullet B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \bullet \overline{B}$	데모르간의 정리

기본원리들은 불표현식을 해석할수 있는 방법을 정의한다. 두개의 분배법칙가운데서 하

나는 초등대수학에서의것과 다르기때문에 주의하여야 한다.

$$A + (B \bullet C) = (A + B) \bullet (A + C)$$

표 I-2의 맨 아래에 있는 두개의 식은 데모르간(DeMorgan)의 정리라고 한다. 이 식들은 다음과 같이 다시 쓸수 있다.

$$A \text{ NOR } B = \overline{A} \text{ AND } \overline{B}$$

$$A \text{ NAND } B = \overline{A} \text{ OR } \overline{B}$$

독자들이 변수 A, B 및 C 에 대하여 실제적인 값(1 및 0)을 대입하는 방법으로 표 I-2의 식들을 검토해 보기 바란다.

제 2 절. 문회로

모든 수자론리회로의 기본조립블로크는 문회로이다. 론리함수는 문회로들의 호상련결에 의하여 실현된다.

문회로는 그의 입력신호들에 대한 간단한 불연산결과를 출력신호로 내보내는 전자회로이다. 수자론리에서 쓰이는 기본문회로에는 AND, OR, NOT, NAND 및 NOR 등이 있다. 그림 I-1 에는 이 5 가지 문회로들을 보여 주었다. 매개 문회로는 도형기호, 대수적표기법 및 진리값의 3 가지 방법으로 정의된다. 여기에서와 부록전반에서 쓰이는 기호는 IEEE 표준(IEEE 표준 91)에 준한것이다. 반전(NOT)연산은 작은 동그라미에 의해 표시된다.

매개 문회로는 하나 또는 두개의 입구와 하나의 출구를 가진다. 입구에서 값이 변화될 때 문회로를 통한 신호는 전달지연시간만큼 지연되여 거의 동시에 정확한 출구로 나간다. 이때의 지연을 문회로지연이라고 한다. 이 문제의 중요성에 대해서는 부록 I 의 제 3 절에서 보게 된다.

그림 I-1 에 보여 준 문회로외에 3, 4 개 또는 그이상의 입구를 가진 문회로도 쓸수 있다. 즉 X+Y+Z 는 3 개의 입구를 가진 OR 문회로로 실현할수 있다.

보통 회로실현에서 모든 문회로류형들이 다 쓰이는것은 아니다. 한두가지 류형의 문회로만 리용하면 설계와 제작이 보다 간단해 진다. 이로부터 문회로의 **함수적완비계**를 정하는것이 중요하다. 이것은 임의의 불함수를 이 완비계에 속하는 문회로만 리용해도 실현할수 있다는것을 의미한다. 다음문회로의 조합은 함수적완비계를 이룬다.

· AND, OR, NOT

· AND, NOT

· OR, NOT

· NAND

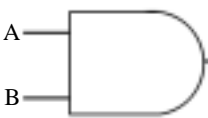
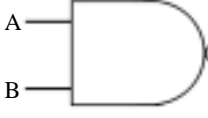
· NOR

이름	도형기호	대수적기능	진리값표
AND	A, B → F	$F = A \bullet B$ 혹은 $F = AB$	A B F 0 0 0 0 1 0 1 0 0 1 1 1
OR	A, B → F	$F = A + B$	A B F 0 0 0 0 1 1 1 0 1 1 1 1
NOT	A → F	$F = \overline{A}$ 혹은 $F = A'$	A F 0 1 0 1
NAND	A, B → F	$F = (\overline{AB})$	A B F 0 0 1 0 1 1 1 0 1 1 1 0
NOR	A, B → F	$F = (\overline{A + B})$	A B F 0 0 1 0 1 0 1 0 0 1 1 0

그림 I-1. 기본론리문회로

AND, OR 및 NOT 문회로는 그것들이 불대수의 3 가지 연산을 표시하므로 함수적완비계를 이룬다는것이 명백하다. AND 와 NOT 문회로가 함수적완비계를 이루자면 AND 와 NOT 연산으로부터 OR 연산을 합성하는 방법이 있어야 한다. 이것은 데모르간의 정리를 적용하면 실현할수 있다. 즉

$$A + B = \overline{\overline{A} \bullet \overline{B}}$$
$$A \text{ OR } B = \text{NOT}((\text{NOT } A) \text{ AND } (\text{NOT } B))$$

마찬가지로 OR와 NOT연산은 그것들을 AND연산을 합성하는데 리용할수 있으므로 함수적완비계를 이룬다.

그림 I-2에는 NAND 문회로 하나만으로 어떻게 AND, OR 및 NOT 함수를 실현하는가를 보여 주었다. 그림 I-3 역시 NOR 문회로를 리용하는 경우 이에 대하여 보여 준다. 이러한 리유로부터 수자회로를 때때로 NAND 문회로나 NOR 문회로 단독으로 구성한다.

문회로를 리해하면 가장 초보적인 준위의 콤퓨터과학 및 콤퓨터공학에 대한 지식을 가지게 된다. 문회로를 구성하는데 리용한 3 극소자결합의 검사는 그 분야와는 다른 전기공학분야에 속하는 문제로 되였다. 그러므로 여기서는 문회로가 수자형콤퓨터의 기본

론리회로를 실현하기 위한 조립블로크로 어떻게 리용될수 있는가를 서술하려고 한다.

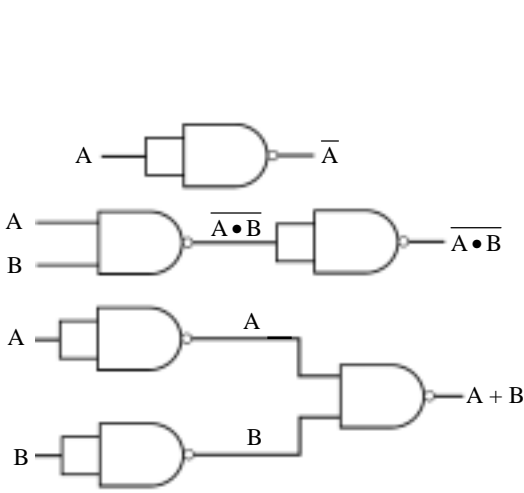



그림 I-2. NAND 문회로의 리용

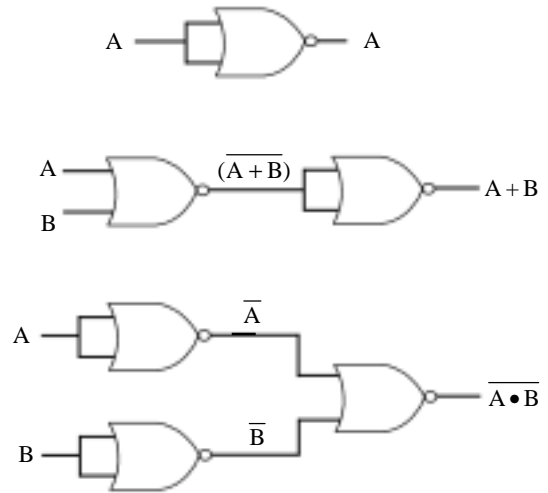



그림 I-3. NOR 문회로의 리용

제 3 절. 조합회로

조합회로는 호상 련결된 문회로들의 묶음으로 된 회로이다. 이 회로의 어떤 순간의 출력신호는 바로 그 순간의 입력신호들만의 함수로 된다. 하나의 문회로에서와 같이 조합회로에 입력신호가 가해 지면 문회로지연만큼 지연될뿐 거의 즉시에 출력신호가 나타난다.

일반적으로 조합회로는 n개의 2진입구와 m개의 2진출구로 이루어 진다. 하나의 문회로와 같이 조합회로는 세가지 방법으로 정의될수 있다.

- **진리값표:** 있을수 있는 2^n 개의 모든 입력신호조합에 대하여 개개의 m 개 출력신호를 2진값으로 표에 목록화한다.
- **도형표시:** 문회로의 호상련결배치를 보여 준다.
- **불식:** 매개 출력신호를 입력신호들에 대한 불함수로 표현한다.

1. 불함수의 실현

임의의 불함수는 문회로의 망과 같은 전기적형태로 실현할수 있다. 임의의 주어 진 함수에 대한 여러가지 실현방도들이 있다. 불함수가 표 I-3의 진리값표에 의하여 제시되였다고 하자. 이제 이 함수를 F가 1이 되게 하는 A, B 및 C의 값들의 조합으로 간단히 항목화하여 표시할수 있다.

$$F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C} \qquad \text{(I-1)}$$

F가 1이 되는 입력값들의 결합에는 3가지가 있으며 이 결합들중 임의의 하나가 일어나면 그 결과는 1로 된다. 이러한 형태의 표시를 **적의 합**(SOP:Sum Of Product)형식이라고 한다. 그림 I-4에는 AND, OR와 NOT 문회로들로 실현한 간단한 형태를 보여 주었다.

표 I-3. 3변수의 불함수

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

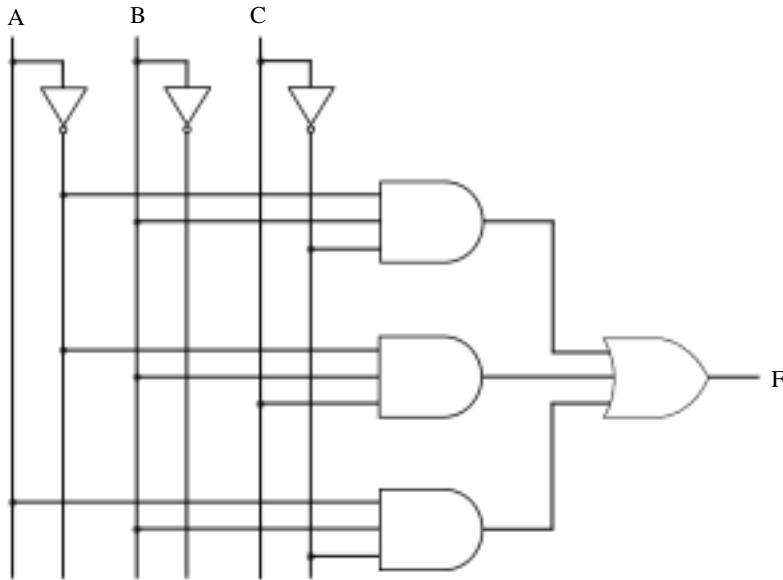



그림 I-4. 표 I-3 의 SOP 실현

다른 형태도 진리값표로부터 얻을수 있다. SOP 형식은 입구조합의 어느것이라도 1 이면 승적항이 참임을 표시한다. 다른 말로 0 을 만드는 입구조합이 없다면 출구가 1 이 된다고 할수 있다. 즉

$$F = \overline{(\bar{A}\bar{B}\bar{C})} \bullet \overline{(\bar{A}\bar{B}C)} \bullet \overline{(A\bar{B}\bar{C})} \bullet \overline{(A\bar{B}C)} \bullet \overline{(ABC)}$$

이것은 데모르간의 정리를 써서 다시 쓸수 있다.

$$\overline{X \bullet Y \bullet Z} = \bar{X} + \bar{Y} + \bar{Z}$$

따라서 F 를 다음과 같이 쓸수 있다.

$$\begin{aligned} F &= (\bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}}) \bullet (\bar{\bar{A}} + \bar{\bar{B}} + \bar{C}) \bullet (\bar{A} + \bar{\bar{B}} + \bar{\bar{C}}) \bullet (\bar{A} + \bar{\bar{B}} + \bar{C}) \bullet (\bar{A} + \bar{B} + \bar{C}) \\ &= (A+B+C) \bullet (A+B+\bar{C}) \bullet (\bar{A}+B+C) \bullet (\bar{A}+B+\bar{C}) \bullet (\bar{A}+\bar{B}+\bar{C}) \qquad (I\text{-}2) \end{aligned}$$

이것을 **합의 적**(POS:Product Of Sum)형식이라고 하며 그림 I-5에 보여 주었다. 그림에서는 잘 안겨 오게 하기 위하여 NOT 문회로는 표시하지 않았다. 그대신 매개 입력신호와 그의 부정을 리용할수 있는것으로 하였다. 이렇게 하면 론리도가 간단해 지며 문회로에서 입력들을 보다 명백하게 읽을수 있다.

이와 같이 불함수는 SOP 또는 POS 형식가운데서 어느 한가지로 실현할수 있다. 두 형식가운데서 어느것을 선택하는가하는것은 진리값표가 출력함수에 대하여 1 혹은 0 중 어느것을 더 포함하는가에 따른다고 볼수 있다. SOP 형식은 매개 1에 대하여 한개의 항을 가지며 POS 형식은 매개 0에 대하여 한개 항을 가진다. 이와는 다른 고찰도 있다.

- SOP 나 POS 보다 일반적으로 진리값표로부터 보다 간단한 불식을 얻어 낼수 있다.
- 단일문회로류형(NAND 또는 NOR)으로 함수를 실현하는것이 더 좋다.

첫번째 점의 의의는 보다 간단한 불식을 쓰면 함수실현에 드는 문회로가 더 적어진다는데 있다. 론리함수를 간단화하는데는 대체로 다음의 3가지 방법을 리용할수 있다.

- 대수적간단화
- 칸도표
- 크와띤-마크라스키표

대수적간단화

대수적간단화는 불식을 보다 적은 요소를 가진 식으로 줄이기 위하여 표 I-2의 항등식의 적용을 동반한다. 실례로 식 I-1을 다시 보기로 하자. 이 식은 일부 공식들을 쓰면 다음식과 등가라는것을 알수 있다.

$$F = \overline{A}B + B\overline{C} \qquad (I\text{-}3)$$

웃식은 다음과 같이 보다 간단한 식으로 된다.

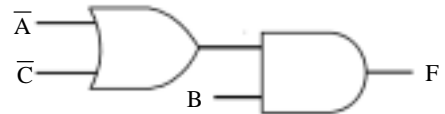
$$F = B(\overline{A} + \overline{C})$$

이 식은 그림 I-6에 보여 준것처럼 실현할수 있다. 식 I-1의 간단화는 기본적으로 관찰에 의하여 진행하였다. 보다 복잡한 식에 대해서는 어떤 보다 체계적인 고찰방법이 있어야 한다.

그림 I-5. 표 I-3의 POS 실현

그림 I-6. 표 A.3 의 간단한실현

칸도표(Karnaugh Map)

칸도표는 간단화의 목적을 위하여 보다 작은 수(4~6)의 변수를 가진 불함수를 표시하는 편리한 방법이다. 도표는 2^n개의 칸들의 배렬로 되여 있는데 그 칸들은 n 개의 2 진변수값들의 있을수 있는 조합을 표시한다. 그림 I-7 ㄱ는 2 변수의 함수인 경우 네개의 칸으로 된 도표를 보여 주고 있다. 앞으로의 목적을 위하여 변수들의 조합을 00, 01,11,10 의 순서로 놓는것이 편리하다. 조합에 대응하는 칸들이 정보를 기록하는데 쓰이므로 조합들은 습관적으로 칸의 웃부분에 표시된

다. 3 변수인 경우에는 표시가 8 개의 칸들의 배렬(그림 I-7 의 ㄴ) 로 되는데 이때 한개 변수에 대한 값은 왼쪽에 놓이고 다른 두개의 변수에 대한 값은 칸우에 놓인다. 4 변수인 경우는 16 개의 칸이 필요하며 그 배치는 그림 I-7 ㄷ에 보여 주었다.

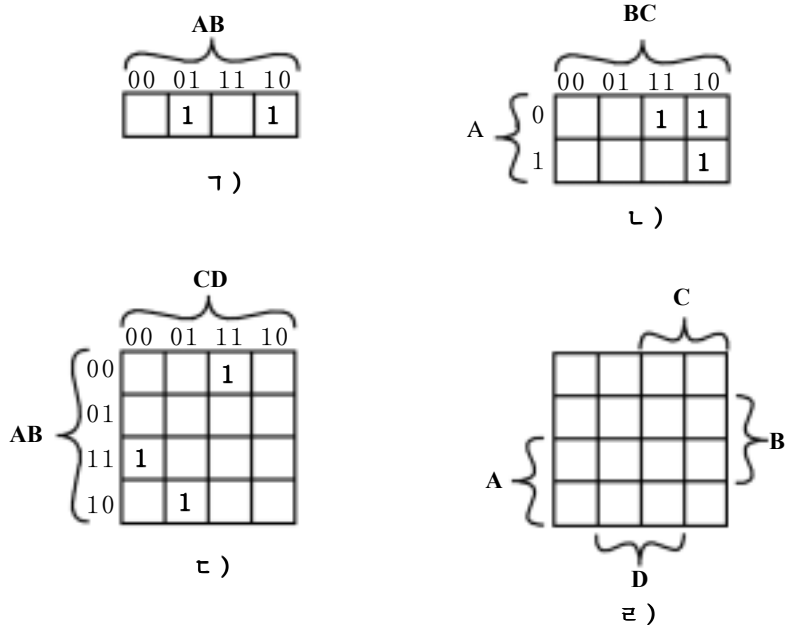



그림 I-7. 불함수를 표현하기 위한 칸도표의 리용

ㄱ－$F = A\overline{B} + \overline{A}B$, ㄴ－$F = \overline{A}B\overline{C} + \overline{A}BC + AB\overline{C}$, ㄷ－

$F = \overline{A}\overline{B}CD + A\overline{B}\overline{C}D + AB\overline{C}\overline{D}$, ㄹ－간단한 표기법

이 칸도표는 다음과 같은 방법으로 어떤 불함수를 표시하는데 쓰인다. 매개 칸들은 적의 합형식에서 고유한 승적항에 대응되며 주어 진 변수에 대해서는 1, 그리고 주어 진 변수의 NOT 에 대해서는 0 을 가진다. 따라서 그림 I-7 ㄱ에서 승적항 $A\overline{B}$ 는 네번째 칸에 해당된다. 함수에서 이러한 모든 승적에 대하여 해당한 칸에 1 을 써넣는다. 그러므로 2 변수실레인 경우 도표는 $A\overline{B} + \overline{A}B$ 에 해당된다. 불함수의 진리값표가 주어 지면 도표를 구성하기 쉽다. 즉 진리값에서 1 의 결과를 만드는 변수값의 매개 조합에 대해서는 도표의 해당한 칸마다 1 을 기입한다. 그림 I-7 ㄴ는 표 I-3 의 진리값표에 대한 결과를 보여 주고 있다. 불식으로부터 칸도표로 바꾸자면 **표준형**이라고 불리우는 형태로 식을 변환하여야 된다. 표준형이 되자면 식에 포함된 모든 항에 매개 변수가 포함되여야 한다. 가령 식 I-3 은 먼저 식 I-1 의 완전한 형태로 확장하고 그다음 도표로 변환한다.

그림 I-7 ㄹ에서 쓰인 변수표시법은 변수들과 도표의 행 및 렬사이의 관계를 강조한 것이다. 여기서 두개의 행은 기호 A 에 의하여 규정되며 거기에 속하는 칸들에서 A 는 값 1 을 가지게 된다. 그리고 기호 A 에 의해 규정되지 못한 나머지 두행에서 A 는 0 이 된다(B, C 및 D 에 대해서도 마찬가지이다.).

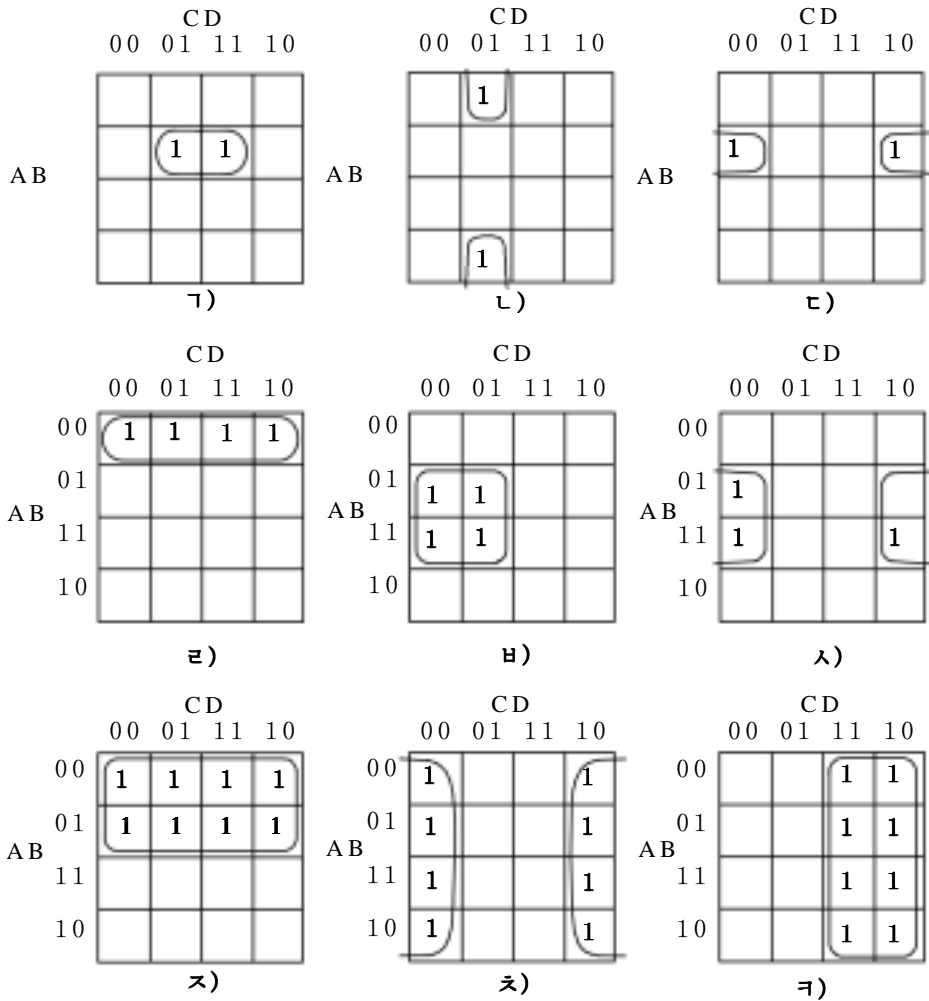



그림 I–8. 칸도표의 리용
ㄱ–ABD, ㄴ–BCD, ㄷ–ABD, ㄹ–AB, ㅁ–BC,
ㅂ–BD, ㅅ–A, ㅇ–D, ㅈ–C

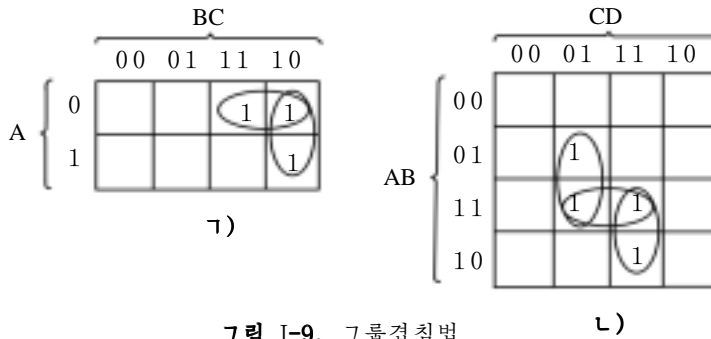
일단 함수의 도표가 작성되면 도표우에 1 의 배렬을 표기하여 간단한 론리식을 얻을 수 있다. 그 원리는 다음과 같다. 서로 린접해 있는 임의의 두칸들은 한개의 변수만이 차이난다. 량쪽이 다 같은 성분을 가지고 있는 두 린접칸인 경우 해당한 승적항들은 오직 한개의 변수만 차이나게 된다. 이런 경우에는 그 차이나는 변수만 제외하고 두항을 합칠수 있다. 가령 그림 I–8 ㄱ에서 두개의 린접칸에 해당되는 두 항은 $\overline{A}B\overline{C}D$ 와 $\overline{A}BCD$ 이다. 이때 함수식을 다음과 같이 쓸수 있다.

$$\overline{A}B\overline{C}D+\overline{A}BCD=\overline{A}BD$$

이 처리는 몇가지 방법으로 확장할수 있다. 우선 린접개념을 도표의 모서리둘레를

묶어서 포함하도록 확장할수 있다. 그러면 한개의 렬에서 맨 꼭대기칸과 맨 아래칸이 린접으로 되며 한행에서 왼쪽과 오른쪽 끝에 있는 칸들이 서로 린접으로 된다. 이와 같은 관계를 그림 I-8 의 ㄴ와 ㄷ에서 설명하고 있다. 다음으로 두개의 칸들만이 아니라 2^n 개의 칸(4,8 등)들도 묶을수 있다. 그림 I-8 에서 다음 3 개의 실례(그림 I-8 의 ㄹ,ㅁ,ㅂ)는 4 개의 칸에 대한 묶기를 보여 주고 있다. 이런 경우에는 두개의 변수들이 제거될수 있다. 마지막 3 개의 실례는 8 개의 칸에 대한 묶기를 보여 준것으로서 이 그룹묶기에서는 3개의 변수들이 제거된다.

이로부터 간단화의 규칙을 다음과 같이 종합할수 있다.

그림 I-9. 그룹겹침법

ㄱ- $F=A\overline{B}+B\overline{C}$, ㄴ- $F=B\overline{C}D+ACD$

- 표식칸(1 을 가진 칸)들가운데서 1, 2, 4 또는 8 개가 되도록 유일한 가장 큰 블로크들을 찾고 그 블로크들을 원형으로 둘러 싼다.
- 될수록 크게 그리고 될수록 개수가 작게 그러면서도 매개 표식칸들이 적어도 한번씩 포함되도록 표식칸들의 보충적인 블로크를 선택한다. 일부 경우에는 결과들이 유일한것이 아닐수 있다. 실례로 어떤 표식칸이 정확히 두개의 다른 칸들과 결합되고 보다 큰 그룹을 완성하기 위한 네번째 표식칸이 없는 경우 두개의 그룹묶음의 선택으로 하나의 선택을 만들수 있게 된다. 그룹을 원형으로 둘러 싸는 경우 같은 1 을 한번이상 리용할수 있다.
- 단일표식칸, 린접표식칸들의 쌍, 4 개 및 8 개 칸들의 그룹 등을 둘러 싸서 원형고리에 들어 가도록 한다. 이와 같은 방법으로 모든 표식칸들이 적어도 하나의 원형고리에 포함되도록 하여 될수록 적은 수의 블로크들이 모든 표식칸들을 포함하도록 한다.

그림 I-9 ㄱ에서는 표 I-3 에 기초하여 이 과정을 보여 주었다. 그룹묶기후에도 고립된다면 1 이 남을수 있다. 이때에는 이것들도 각각 1 의 그룹으로 보고 원형으로 둘러 싼다. 마지막으로 도표로부터 간단화된 불식으로 가기전에 다른 그룹에 의해 완전히 겹쳐진 임의의 1 의 그룹은 제외시킬수 있다. 이것을 그림 I-9 ㄴ에 보여 주었다. 이 경우 수평방향묶음이 중복부분으로 되며 불식을 작성할 때 무시된다.

칸도표의 또 하나의 특징에 대해서도 언급할 필요가 있다. 일부 경우에는 변수의 어떤 값조합이 결코 생기지 않으며 따라서 그에 대응한 출력도 결코 생기지 않는다. 이것을 《무이금지항》조건이라고 한다. 이런 조건을 가진 매항에 대해서는 도표의 해당한

칸에 문자《d》를 써 넣는다. 그룹묶기와 간단화를 진행할 때 매개《d》는 어느 쪽이 가장 간단한 식으로 되든지간에 1 또는 0 으로 취급할수 있다.

표 I-4. 한자리조임형 10 진수 하나증가에 대한 진리값표

번호	A	B	C	D	번호	W	X	Y	Z
0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	2	0	0	1	0
2	0	0	1	0	3	0	0	1	1
3	0	0	1	1	4	0	1	0	0
4	0	1	0	0	5	0	1	0	1
5	0	1	0	1	6	0	1	1	0
6	0	1	1	0	7	0	1	1	1
7	0	1	1	1	8	1	0	0	0
8	1	0	0	0	9	1	0	0	1
9	1	0	0	1	0	0	0	0	0
조건에 무관계	1	0	1	0		d	d	d	d
	1	0	1	1		d	d	d	d
	1	1	0	0		d	d	d	d
	1	1	0	1		d	d	d	d
	1	1	1	0		d	d	d	d
	1	1	1	1		d	d	d	d

이 문제는 [HAYE88]에서 제시된 실례에서 설명해 주고 있다. 이제 조임형 10 진수에 1을 더하는 회로에 대한 불식을 개발한다고 하자. 제 9 장 제 2 절로부터 조임형 10 진수에서는 매개 10 진수자를 명시적인 방법으로 4bit 의 코드로 표시한다는것을 알수 있다. 즉 0=0000, 1=0001, •••, 8=1000 및 9=1001 이 된다. 1010 부터 1111 까지의 나머지 4bit 값들은 쓰이지 않는다. 이 부호는 2 진식 10 진 (BCD)부호라고도 한다.

표 I-4 는 4bit BCD 입구보다 하나이상인 4bit 결과를 주는 진리값표를 보여 주고 있다. 더하기는 10 진수더하기(모듈 10)이며 따라서 9+1=0 이다. 또한 6 개의 입구부호는 있을수 없는 BCD 입구이므로《무이금지항》결과를 만들어 낸다. 그림 I-10 은 얻어진 매개 출구변수에 대한 칸도표를 보여 주고 있다. 여기서 d 칸들은 가장 좋은 가능한 그룹묶기를 수행하는데 쓰이였다.

크와인-마크라스키법

칸도표법은 변수가 4 개이상인 경우 매우 쓰기 불편하다. 5 개의 변수에서는 두개의 16×16 칸도표가 요구되며 하나의 도표는 린접성을 달성하기 위하여 3 차원에서 다

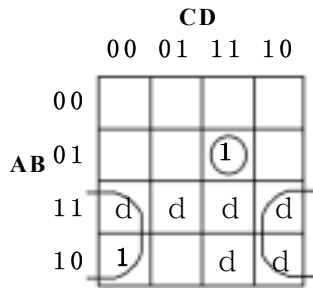
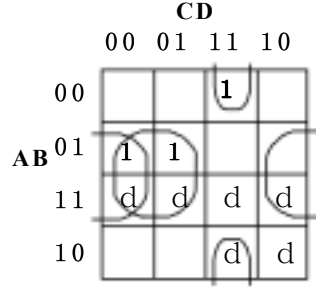
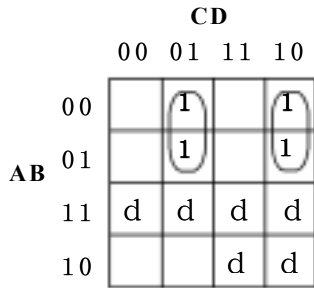
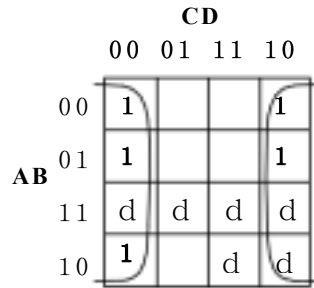



그림 I-10. 칸도표의 증가법

ㄱ - $W = \overline{A}D + \overline{A}BCD$, ㄴ - $X = BD + \overline{B}C + BCD$,

ㄷ - $Y = \overline{ACD} + \overline{A}C\overline{D}$, ㄹ - $Z = \overline{D}$

른 도표의 꽉대기에 있는것으로 간주해야 한다. 6 변수에 대해서는 4 차원에서 4 개의 16 ×16 칸도표를 리용하여야 한다. 칸도표를 대신하는것으로는 크와인-마크라스키(Quine-McKluskey)법이라고 불리우는 표에 의한 방법이 있다. 이 방법은 최소화된 불식을 작성하기 위한 자동도구를 주므로 콤퓨터에서의 프로그람작성에도 어울린다.

이 방법은 한가지 실례에 의해서도 잘 설명할수 있다. 이제 다음과 같은 식을 보기로 하자.

$$ABCD + AB\overline{C}D + AB\overline{CD} + A\overline{B}CD + \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{ABC}D$$

우의 식이 어떤 진리값표로부터 유도되였다고 하자. 이 식을 문회로로 실현하는데 알맞는 최소론리식으로 만들려고 한다.

첫 단계에서는 표를 구성하는데 이때에는 표의 매개 렬을 론리식의 승적항의 하나에 해당되도록 한다. 항들은 부정변수의 개수에 따라 묶어 진다. 즉 부정이 없는 항부터 시작하여 그러한 항이 있다면 한개의 부정변수를 가진 모든 항을 묶는식으로 해나간다. 표 I-5에서는 실례로 든 론리식의 목록을 보여 준다. 표에서 수평선들은 그룹묶기를 표시하는데 쓰이였다. 명백한바와 같이 매개 항은 부정이 아닌 변수에 대해서는 1, 부정변수에 대해서는 0 으로 표시하였다. 즉 그것들이 포함한 1 의 개수에 따라 항들을 묶었다. 첨수렬은 앞으로 써먹게 되는 10 진등가수를 그냥 첨부한것이다.

다음단계는 한개의 변수만 차이나는 항들의 모든 쌍(항들중 한개 항에서 한 변수가 0 이고 다른 세개는 1 인것을 내놓고는 같은 그런 항들의 모든 쌍)을 찾아 내는것이다. 이 방법에서는 항들을 그룹화하였기때문에 첫 그룹으로부터 시작하여 그 그룹의 매개 항을 두번째 그룹의 매개 항과 비교하는 방법으로 진행할수 있다. 다음에는 두번째 그룹의 매개 항을 세번째 그룹의 모든 항들과 비교한다. 이러한 방법으로 정합이 발견될 때마다 매개 항의 뒤에 검사표식을 주며 두 항에서 차이나는 변수를 제거함으로써 쌍을 묶고 그

표 I-5. 크와인-마크라스키방법의 첫 상태

(실례로 $F = ABCD + AB\bar{C}D + AB\overline{CD} + A\bar{B}CD + \bar{A}BCD + \bar{A}BC\bar{D} + \bar{A}B\bar{C}D + \overline{ABC}D$)

결과식	**첨수**	**A**	**B**	**C**	**D**	
$\overline{ABC}D$	1	0	0	0	1	√
$\bar{A}B\bar{C}D$	5	0	1	1	1	√
$\bar{A}BC\bar{D}$	6	0	1	1	0	√
$AB\overline{CD}$	12	1	1	0	0	√
$\bar{A}BCD$	7	0	1	1	1	√
$A\bar{B}CD$	11	1	0	1	1	√
$AB\bar{C}D$	13	1	1	0	1	√
ABCD	15	1	1	1	1	√

것을 새로운 목록에 첨가한다. 실례로 ABCD 와 ABCD 항들은 승적항 ABC 로 결합된다. 이 과정을 전체 원시표가 모두 검사될 때까지 계속한다. 결과는 다음과 같은 기입(입구)들을 가진 새로운 표로 된다.

$\overline{AC}D$	$AB\bar{C}$	ABD √
	$B\bar{C}D$ √	ACD
	$\bar{A}BC$	BCD √
	$\bar{A}BD$ √	

이 새로운 표는 첫표와 같은 양식으로 그룹화된다. 두번째 표는 첫번째 표와 같은 수법으로 작성된다. 즉 한 변수만 차이나는 항들이 검사되여 세번째 표를 위한 새로운 항들이 만들어 진다.

보통 이 프로쎄스는 정합이 없는 표가 만들어 질 때까지 표들에 대하여 련속 진행된다. 우리의 경우에는 이 프로쎄스에 3 개의 표가 포함된다.

일단 정확히 작성된 프로쎄스가 완결되면 식의 많은 가능한 항들이 소거된다. 소거되지 않은 항들은 표 I-6 에 보여 준것과 같이 행렬을 구성하는데 쓰인다. 행렬의 매개 행은 지금까지 표에서 쓰인 소거되지 않은(검토가 안된) 항들중 어느 하나에 해당한다. 매개 렬은 본래의 식에 있던 항들중 어느 하나에 해당한다. 렬요소와 《량립성》인 행요소들에 대해서는 행과 렬의 매개 사귐점에 X 가 기입된다. 즉 행요소에 있는 변수가 렬요소에 있는 변수와 같은 값을 가진다. 다음으로 렬에서 홀로 있는 X 표식에 동그라미를

친다. 그다음 동그라미를 친 X 표식이 있는 임의의 행에서 모든 X 표식에 직 4 각형을 친다. 매개 렬이 직 4 각형을 친 X 나 동그라미를 친 X 표식을 가지고 있다면 X 표식을 가

표 I-6. 크와인-마크라스키방법의 첫 상태

	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD	ABCD
BD	X	X			X		X	
ACD							[X]	⊗
ABC					[X]	⊗		
ABC		[X]	⊗					
ACD	[X]			⊗				

(실례로 $F = ABCD + AB\overline{C}D + AB\overline{CD} + A\overline{B}CD + \overline{A}BCD + \overline{A}BC\overline{D} + \overline{A}B\overline{C}D + \overline{ABC}D$)

진 행요소들이 최소식을 구성한다. 결국 이 실례에서 마지막식은 다음과 같다.

$$AB\overline{C} + ACD + \overline{A}BC + \overline{AC}D$$

동그라미도 직 4 각형도 못 가진 렬이 있는 경우에는 보충적인 조작이 요구된다. 본질적으로는 모든 렬에 X 표식이 기입될 때까지 행요소를 보충하게 된다.

이제 왜 그렇게 되는가를 직관적으로 정당화하기 위하여 크와인-마크라스키법을 요약해 보자. 조작의 첫 단계는 상당히 간단하다. 이 프로쎄스는 승적항에서 불필요한 변수를 제거한다. 즉 $ABC + AB\overline{C}$ 라는 식은

$$ABC + AB\overline{C} = AB(C + \overline{C}) = AB$$

이므로 AB 와 등가이다.

변수를 제거한 다음에는 본래식과 명백히 등가인 식이 얻어 진다. 그러나 이 식에도 칸도표에서 여분의 항에 대한 그룹묶기를 한것처럼 여분의 항이 있을수 있다. 행렬지면배치는 본래의 식에서 매개 항이 보존되고 최종식에서 항의 개수가 최소로 되는 방법으로 간단화한다.

NAND 및 NOR 실현

불함수의 실현에서 다른 고려사항은 사용되는 문회로의 종류와 관계된다. 불함수를 NAND 문회로 또는 NOR 문회로만으로 실현하는 문제가 자주 제기된다. 이렇게 하는것이 최소문회로실현으로 되지 않을수도 있지만 이것은 규격화라는 우점을 가지며 제조공정을 간단화할수 있다. 다시 식 I-3 을 보기로 하자.

$$F = B(\overline{A} + \overline{C})$$

어떤 값의 부정의 부정은 바로 그 본래의 값과 같으므로 웃식은 다음과 같다.

$$F = B(\overline{A} + \overline{C}) = (\overline{A}B) + (B\overline{C})$$

데모르간의 정리를 적용하면

$$F = (\overline{A}B) \bullet (B\overline{C})$$

로 되며 결국 그림 I-11 에 보여 준것과 같이 이 함수는 3 개의 NAND 문회로로 실현된다.

그림 I-11. 표 I-3 의 NAND 실현

D0
D1
D2
D3
4-1 선
다중선택기
(MUX)
F
S2 S1

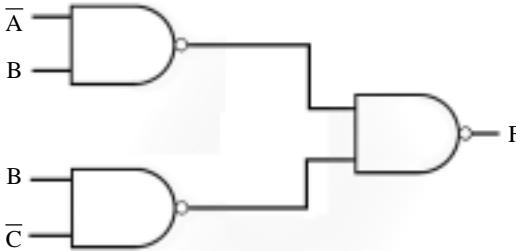
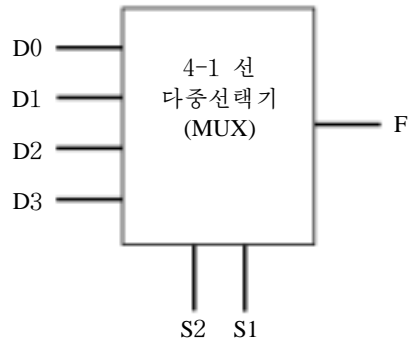



그림 I-12. 4 입구 1 출구다중선택기표현

2. 다중통로기

다중통로기는 다중입구를 하나의 출구에 이어 주는 회로이다. 어떤 순간에 한개의 입력신호가 출구에 나가도록 선택된다. 일반적인 구성도를 그림 I-12 에 보여 주었다. 이것은 4-1 다중통로기를 표시하고 있다. 4 개의 입력신호선은 D0, D1, D2 및 D3 이라고 한다. 이 신호선가운데서 하나가 선택되여 출력신호 F 를 만든다. 4 개의 입구가운데서 하나를 선택하기 위하여 2bit 선택부호가 요구되며 이것은 S1 및 S2 라고 하는 두개의 선택신호선에 의하여 실현된다.

표 I-7. 4-1 다중통로기의 진리값표

S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

실례로 든 4-1 다중통로기는 표 I-7 의 진리값표에 의하여 정의된다. 이것은 진리값표의 간단화된 형식이다. 입력변수의 있을수 있는 모든 조합대신에 출구를 신호선 D0, D1, D2 및 D3 으로부터의 자료로 보여 주고 있다. 그림 I-13 에는 AND, OR 및 NOT 문회로를 써서 구성한 회로를 보여 주었다. S1 과 S2 는 그것들의 임의의 조합에 대하여 3 개의 AND 문회로의 출력값이 0 이 되도록 AND 문회로에 련결된다. 이때 네번째 AND 문회로는 선택된 신호선으로 0 또는 1 값을 내보낸다. 결국 OR 문회로의 3 개의 입구는 언제나 0 이고 OR 문회로의 출구로는 선택된 입구문회로의 값이 나가게 된다. 이런 규격화된 구성을 리용하면 8-1, 16-1 등의 각이한 형식의 다중통로기를 구성하기가 쉽다.

다중통로기는 신호와 자료의 경로를 조종하는 수자회로에서 쓰인다. 한가지 실례로는 프로그람계수기(PC)의 넣기회로를 들수 있다. 이때 프로그람계수기에 설정되는 값은

여러개의 각이한 원천들가운데서 어느 하나로부터 오게 할수 있다.

- 다음명령을 위하여 PC 의 내용을 하나 증가시키는 경우는 2 진계수기
- 직접주소를 써서 갈래명령을 실행하는 경우는 명령등록기
- 변위방식을 써서 갈래명령이 주소를 지정하는 경우는 ALU 의 출구

이 각이한 입구들은 다중통로기의 입력신호선에 련결하며 그 출구는 PC 에 련결된다. 선택신호선은 PC 에 넣어 질 값을 결정한다. PC 는 여러개의 비트로 구성되여 있으며 매 비트마다 여러개의 다중통로기가 쓰인다. 그림 I-14 에는 16bit 주소인 경우 이것을 보여 주었다.

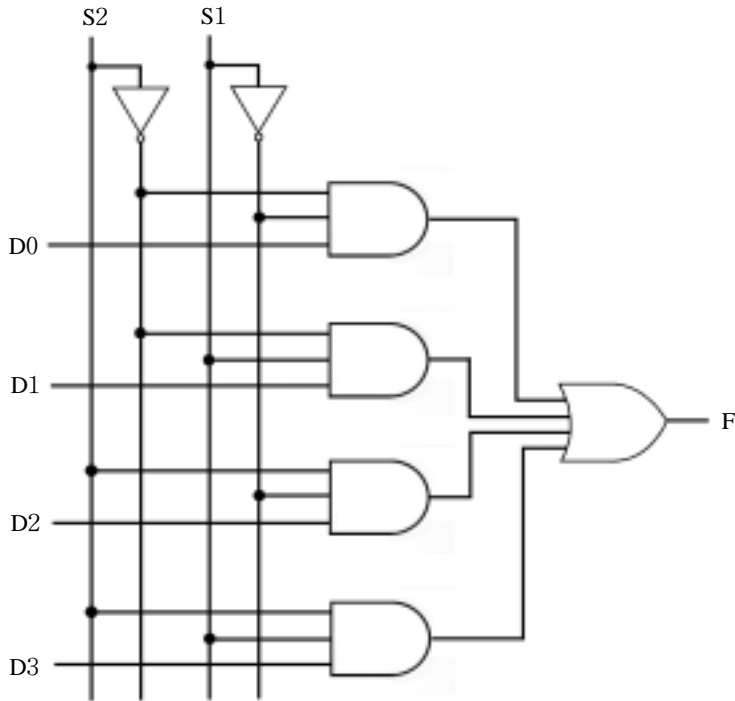



그림 I-13. 다중선택기표현

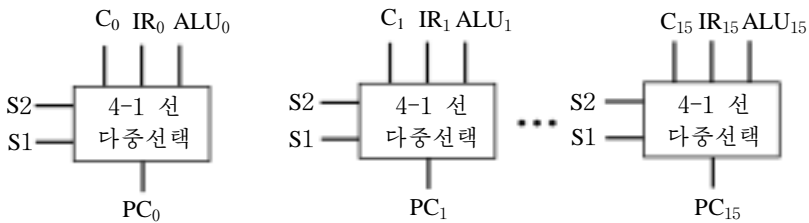



그림 I-14. 프로그람계수기에 다중선택기입력

3. 해신기

해신기는 입력신호선의 패턴에 따라 어떤 순간에 여러개의 출력신호선가운데서 하나

에만 출력신호가 나오게 된 조합회로의 한 종류이다. 해신기는 보통 n 개의 입구와 2n 개의 출구를 가지고 있다. 그림 I-15 에는 3 입구, 8 출구를 가진 해신기를 보여 주었다.

해신기는 수자형콤퓨터에서 많이 쓰인다. 한가지 실례는 주소해신이다. 256 × 8bit RAM 소자 4 개를 써서 1Kbyte 의 기억기를 구성한다고 하자. 이때 다음과 같이 분할되는 유일적인 하나의 주소공간을 형성하려고 한다.

주소	소편
0000 − 00FF	0
0100 − 01FF	1
0200 − 02FF	2
0300 − 03FF	3

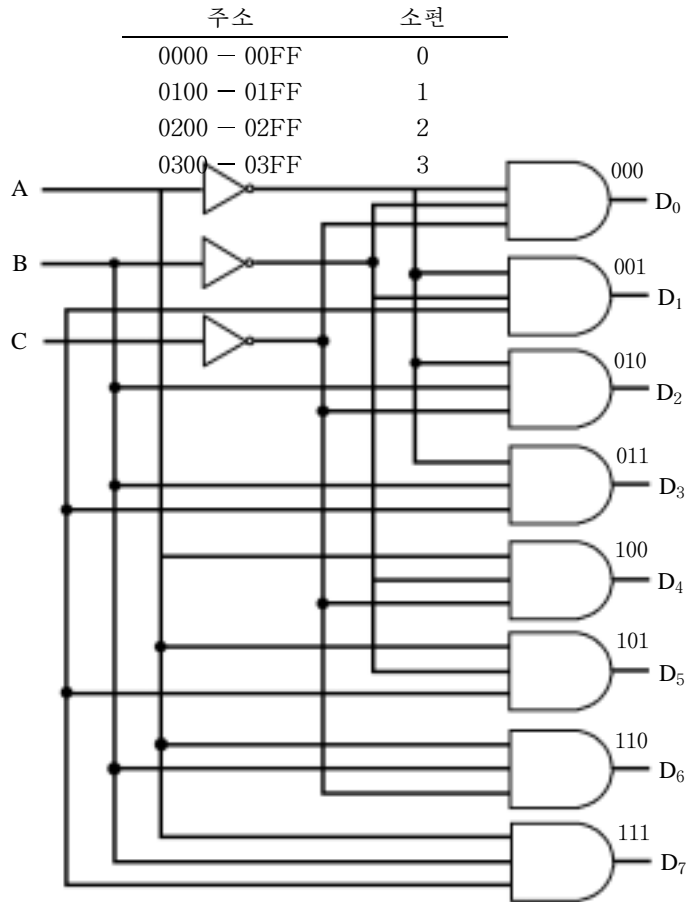



그림 I-15. 3 입구 2^3=8 출구해신기

매개 소자는 8 개의 주소선을 요구하며 이것들은 낮은 자리 8bit 주소에 의하여 신호가 공급된다. 10bit 주소가운데서 높은자리 2bit 는 4 개의 RAM 소자가운데서 하나를 선택하는데 리용된다. 이 목적을 위하여 2-4 해신기는 그림 I-16 에서 보여 준바와 같이 4 개의 소자중 어느 하나의 출구를 가능하게 하는데 리용된다.

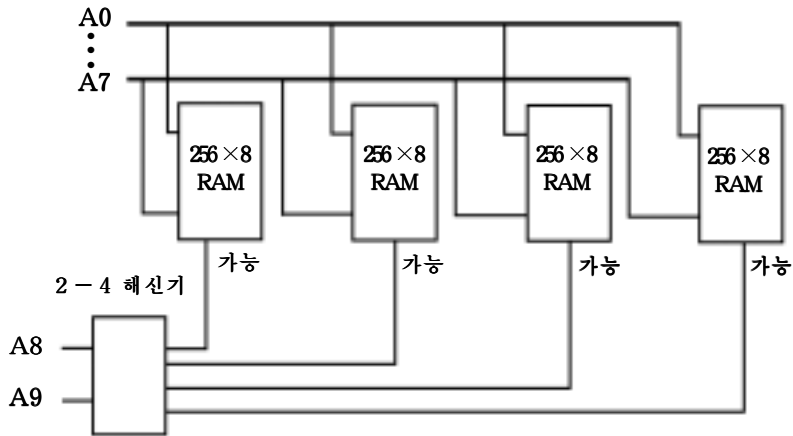



그림 I-16. 주소해신

해신기는 보충적인 입력신호선을 가지면 분배기로 쓸수 있다. 분배기는 다중통로기의 반대기능을 수행하며 이 회로는 한개의 입구를 여러개의 출구가운데서 하나에 이어주게 된다. 이 회로를 그림 I-17에 보여 주었다. 먼저 2^n개의 첫 출력신호가운데서 하나를 만들기 위하여 n 개의 입력신호가 해신된다. 2^n개의 모든 출력신호선들은 자료입력신호선과 론리곱하기되여 있다. 따라서 어떤 출력신호선을 선택하기 위하여 주소로서 n 개의 입구가 작용한다. 그리고 자료입력신호선의 값(0 또는 1)이 출력신호선에 이어 진다.

그림 I-17 에서 구조를 다른 방법으로 고찰해 볼수 있다. **자료입력으로부터 가능**에로의 새로운 선에 대한 표식을 변화시키자. 이것은 해신기의 시간일치조종을 가능하게 한다. 해신출구는 해신입구가 있고 가능선이 1 의 값을 가질 때에만 나타난다.

그림 I-17. 비다중선택기를 리용한 해신기실행

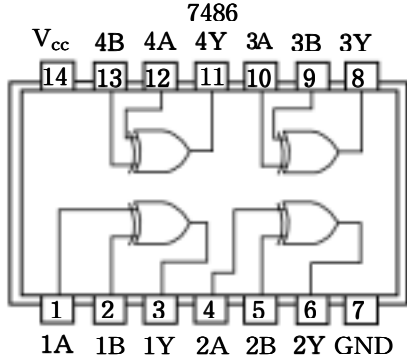
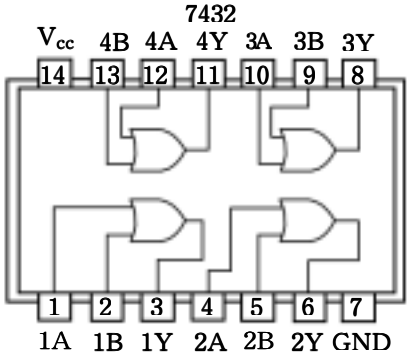
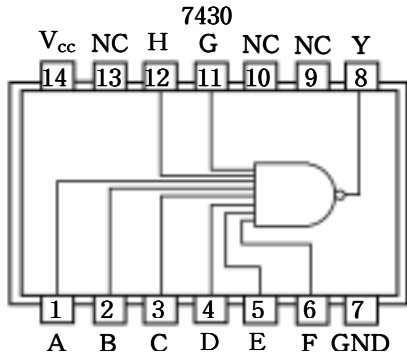
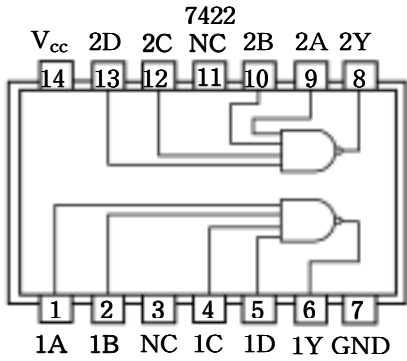
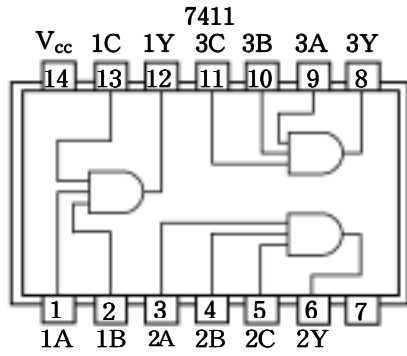
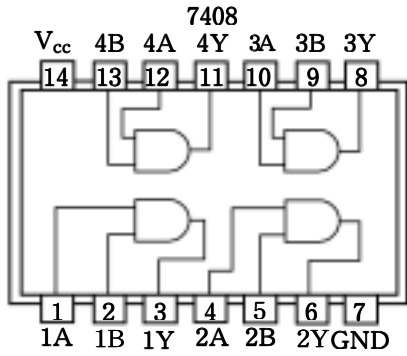
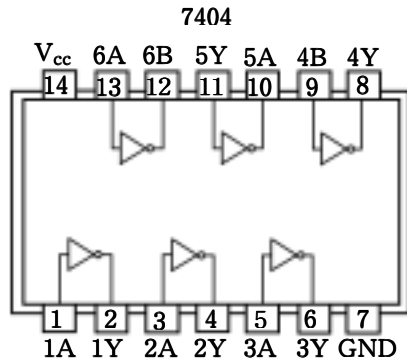
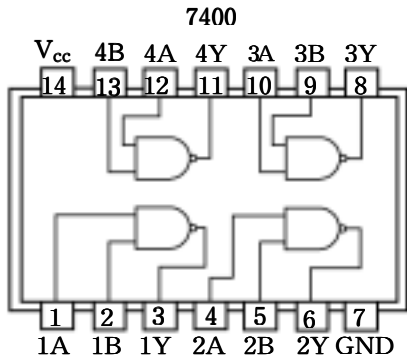



그림 I-18. SSI 의 소편들의 단자배치 (설계공학자들을 위한 TTL 자료집 copyright © 1976 Texas Instrument 회사)

4. 프로그람식론리배렬

지금까지는 임의의 기능을 실현할수 있는 조립블로크로서 개별문회로들을 론의하였다. 설계자들은 해당한 불식을 조작하여 사용되는 문회로의 수를 최소화하는 전략을 추구하였다.

집적회로에 의하여 제공되는 집적도준위의 강화로 다른 고려가 적용된다. 소규모집적회로(SSI)를 쓴 초기의 집적회로는 한 소편우에 1～10 까지의 문회로를 집적되였다. 매개 문회로들은 지금까지 서술된 조립블로크방법으로 따로따로 취급된다. 그림 I-18 에는 일부 SSI 소자들의 실례를 주었다. 론리함수를 구성하기 위하여 많은 소자를 인쇄회로기판우에 배치하고 적당하게 단자들을 호상 련결한다.

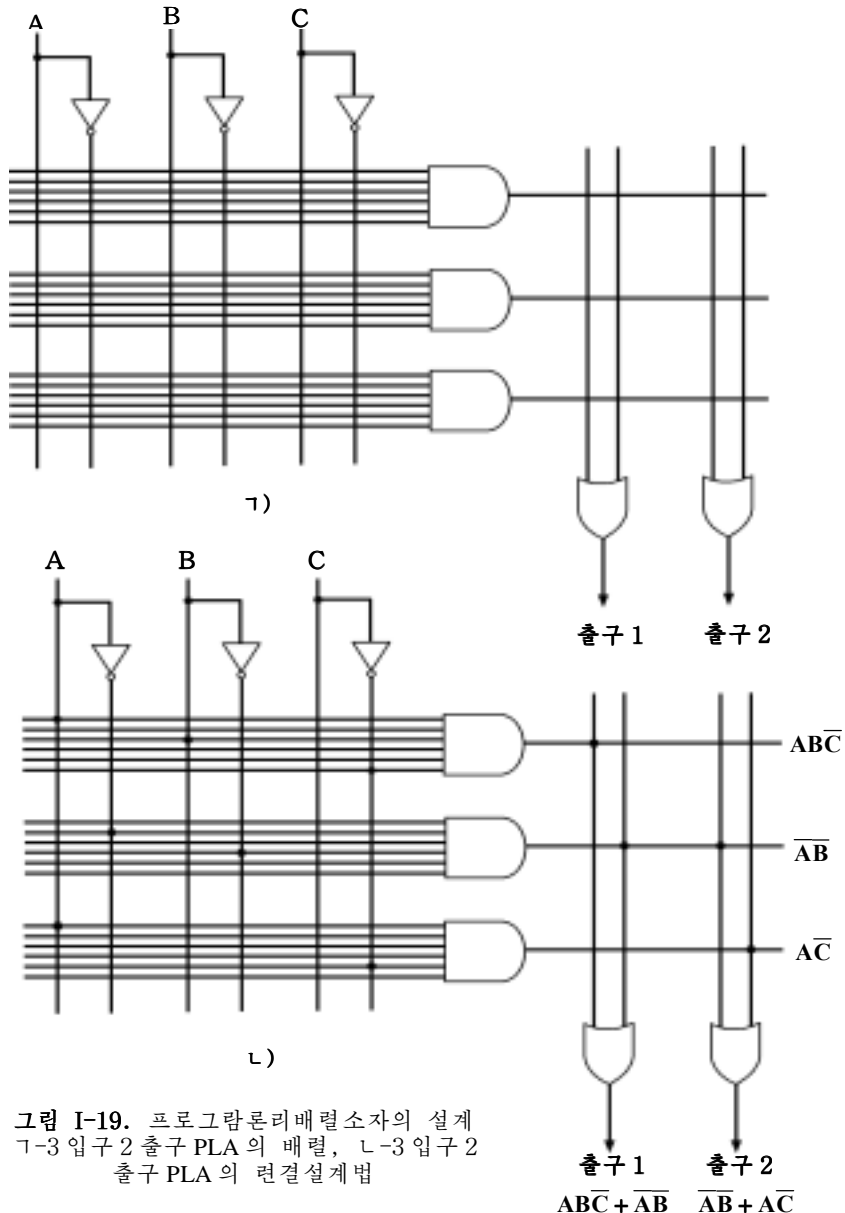



그림 I-19. 프로그람론리배렬소자의 설계
ㄱ-3 입구 2 출구 PLA 의 배렬, ㄴ-3 입구 2 출구 PLA 의 련결설계법

집적도의 수준의 높아 지면서 한 소편우에 보다 많은 문회로를 배치할수 있게 되였을뿐아니라 그 소편에서 문회로의 호상 련결을 잘 할수 있게 되였다. 이것은 가격저하, 크기감소 및 동작속도증가(보통 소편내에서의 지연이 소편밖에서의 지연보다 더 짧으므로) 등의 우점을 가져 온다. 그러나 설계에서 문제가 생긴다. 매개의 특수한 론리함수 또는 함수의 묶음에 대하여 소편우에서의 문회로의 배치와 호상련결을 설계하여야 한다. 이러한 전용소자설계에 동반되게 비용과 시간이 많이 든다. 그리하여 특수목적에 실지로 적응할수 있는 일반목적소편을 개발하는것이 관심사로 되였다. 그 관심을 모으고 있는것이 **프로그람식론리배렬**(PLA:**Programmable Logic Array**)소자이다.

PLA 는 이미 본 임의의 불함수(진리값표)를 적의 합(SOP)형식으로 표현할수 있다는 사실에 기초하고 있다. PLA 는 한 소편우에 있는 NOT, AND 및 OR 문회로의 규칙적인 배렬로 이루어 진다. 매개 소편입구는 매개 입구와 그의 부정을 매 AND 문회로에 가해 지도록 NOT 문회로를 통하여 지나가게 되여 있다. 매 AND 문회로의 출구는 매개 OR 문회로에 가해 지며 다시 매개 OR 문회로의 출구는 소자의 출구로 된다. 적당히 련결하면 임의의 SOP 식을 실현할수 있다.

그림 I-19 ㄱ에는 3개 입구와 8개의 문회로 및 두개의 출구를 가진 PLA 를 보여 주었다. 대부분의 대규모 PLA 는 수백개의 문회로와 15~25 개의 입구 및 5~15 개의 출구를 가지고 있다. 입구로부터 AND 문회로에로의 련결, AND 문회로로부터 OR 문회로의 련결은 규정되여 있지 않다.

PLA 는 프로그람작성(련결의 작성)을 쉽게 하도록 두가지 서로 다른 방법으로 만든다. 첫번째 방법에서는 가능한 모든 련결을 그 련결점에 휴즈를 집어 넣어 만든다. 불필요한 련결은 휴즈를 끊어 버리여 제거할수 있다. 이런 종류의 PLA 를 **사용자 PLA** 라고 한다. 한편 두번째 방법에서는 알맞는 호상련결패턴으로 공급되는 적당한 마스크를 리용하여 소편제조과정에 정확한 련결이 이루어 진다. 이 두 경우에 PLA 는 융통성이 있고 값 눅은 수자론리함수를 실현할수 있게 한다.

그림 I-19 ㄴ에는 두 불식을 실현하는 설계를 보여 주었다.

5. 읽기전용기억기

조합회로는 그의 출력이 오직 현재의 입력신호에 따라서 결정되며 이전 입력신호들의 경력에는 무관계하므로 흔히 《무기억》회로로 간주한다. 그러나 조합회로로 실현되는 한가지 부류의 기억기인 **읽기전용기억기**(ROM)가 있다.

ROM 이 읽기동작만을 수행하는 기억기라는것을 상기해 보자. 이것은 ROM 에 기억된 2진정보가 제조공정기간에 작성되여 영구적이라는것을 의미한다. 따라서 ROM 에 가해 진 입력신호(주소선)는 늘 같은 출력신호(자료선)를 만든다. 출력이 현재 가해 지는 입력신호의 함수로 되기때문에 ROM 은 사실상 조합회로이다.

ROM 은 해신기와 OR 문회로의 묶음으로 실현할수 있다. 실례로 표 I-8 을 보기로 하자. 이것은 4 개의 입구와 4 개의 출구를 가진 진리값표로 볼수 있다. 가능한 16 개의 매개 입구변수에 대응하는 묶음의 출력값들을 표에 보여 주고 있다. 이것은 역시 각각 4bit 인 16 단어로 된 64bitROM 의 내용을 정의한것으로 볼수 있다. 여기서 4 개의 입구는 주소를 지정하며 4 개의 출구는 그 주소에 의해 지정된 위치의 내용을 지정한다. 그림 I-20에는 이 기억기를 4-16해신기와 4개의 OR문회로를 사용하여 어떻게 실현하는가를 보여 주었다. PLA 와 같은 규칙적인 구성을 리용하여 호상련결을 요구하는 결과를 반영하도록 실현한다.

표 I-8. ROM에 대한 진리값표

입구				출구			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

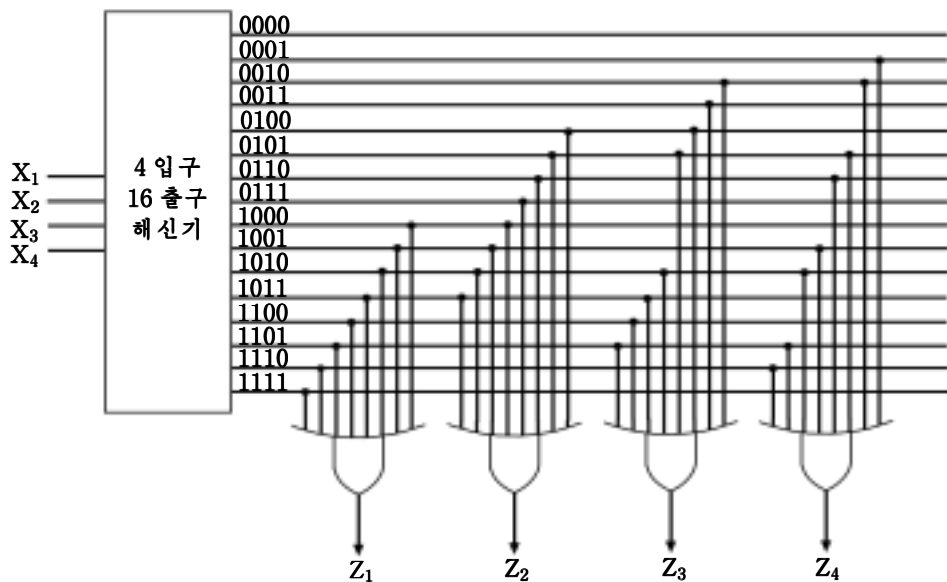



그림 I-20. 64bitROM

7. 가산기

지금까지는 호상 련결된 문회로들이 신호의 경로조종, 해신, ROM과 같은 기능을 실현하는데 어떻게 쓰이는가를 보았다 .아직까지 보지 못한 하나의 기본령역은 산수연산이다. 여기서는 더하기기능에 대하여 간단히 개괄한다.

2진수더하기는 불대수와 다르며 여기서는 결과에 자리올림항이 포함된다.

$$\begin{array}{r}0\\+\ 0\\\hline 0\end{array}\qquad\begin{array}{r}0\\+\ 1\\\hline 1\end{array}\qquad\begin{array}{r}1\\+\ 0\\\hline 1\end{array}\qquad\begin{array}{r}1\\+\ 1\\\hline 10\end{array}$$

그러나 더하기를 불항과 관계시킬수 있다. 표 I-9 ㄱ에는 1bit의 합과 자리올림비트를 만들기 위하여 두개의 입구비트를 더하는 론리를 보여 주었다. 이 진리값표는 수자론리로 쉽게 실현된다. 그러나 꼭 한쌍의 비트에 대한 더하기를 수행하는데는 흥미가 없다. 요구되는것은 두개의 n-bit 수에 대한 더하기이다. 이 문제는 한 가산기로부터의 자리올림을 다음가산기에 입력으로 제공하는 식으로 가산기의 묶음을 만들면 풀수 있다. 그림 I-21에는 4bit 가산기를 보여 주었다.

표 I-9. 2진더하기의 진리값표

ㄱ) 단일비트더하기

A	B	합	자리올림
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

ㄴ) 자리올림을 고려한 더하기

C_{in}	A	B	합	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

여러 비트가산기가 동작하도록 하자면 매 단일비트가산기가 아래자리로부터의 자리올림까지 포함하여 3개의 입구를 가져야 한다. 수정된 진리값을 표 A-9 ㄴ에 주었다. 두개의 출력은 다음식으로 표시된다.

$$\mathrm{Sum} = \overline{AB}C + \overline{A}B\overline{C} + ABC + A\overline{BC}$$
$$\mathrm{Carry} = AB + AC + BC$$

그림 I-22에는 AND, OR 및 NOT 문회로를 쓴 가산기의 실현을 보여 주었다.

결국 그림 I-23에 보여 준것과 같은 다중비트가산기를 실현하기 위한 필요한 론리회로를 가지게 된다. 매개 가산기로부터의 출력은 앞단가산기로부터의 자리올림에 관계되므로 맨 아래자리부터 맨 웃자리까지의 지연은 늘어 난다.

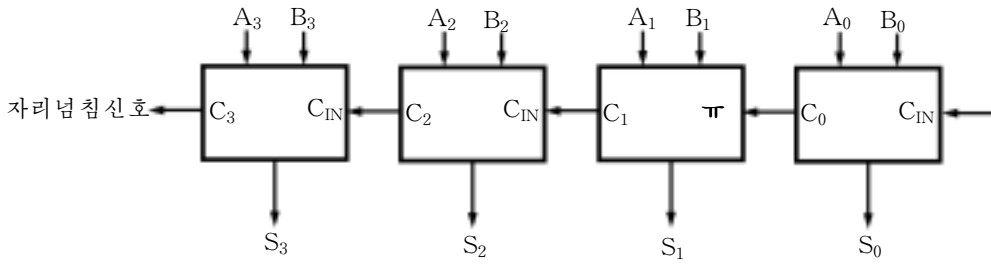



그림 I-21. 4bit 가산기

매개 단일비트가산기는 일정한 크기의 문회로지연을 가지며 이 문회로지연은 루적된다. 비트수가 큰 가산기에 대해서는 이 루적된 지연이 허용할수 없이 큰 값이 될수 있다.

만일 자리올림값이 모든 앞단을 통하여 절단되는것이 없이 결정된다면 매개 단일비트가산기는 독립적으로 작용하며 지연도 루적되지 않는다. 이것은 **자리올림미리보기**로 알려 진 수법에 의하여 얻어 질수 있다. 이 수법을 설명하기 위하여 4bit 가산기를 다시 보기로 하자.

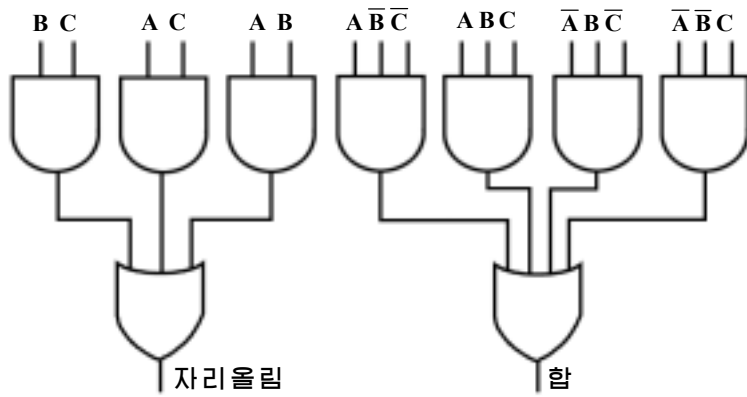



그림 I-22. 가산기의 실현

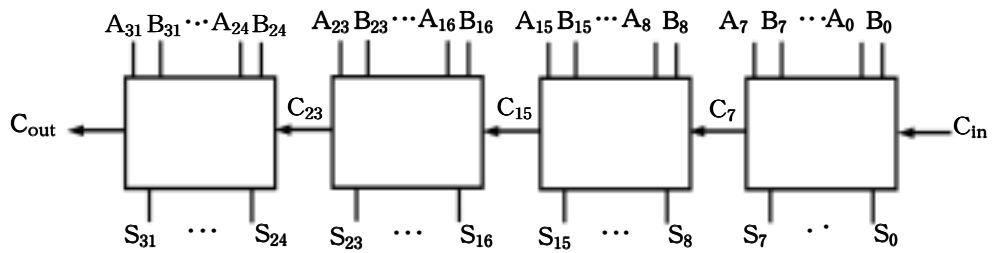



그림 I-23. 8bit 가산기를 리용한 32bit 가산기구성

선행자리올림값에 대한 고려가 없이 임의의 단의 가산기에 대한 자리올림입구를 지정하는 식을 생각하면 그것은 다음과 같다.

$$C_0 = A_0 B_0 \qquad (I-4)$$

$$C_1 = A_1 B_1 + A_1 A_0 B_0 + B_1 A_0 B_0 \qquad (I-5)$$

같은 절차로 따져 보면 다음식을 얻는다.

$$C_2 = A_2 B_2 + A_2 A_1 B_1 + A_2 A_1 A_0 B_0 + A_2 B_1 A_0 B_0 + B_2 A_1 B_1 + B_2 A_1 A_0 B_0 + B_2 B_1 A_0 B_0$$

이 과정을 임의의 긴 가산기에 대하여서도 적용할수 있다. 매개 자리올림항은 자리올림에 무관계하게 본래의 입력에만 관계되는 함수로서 SOP 형식으로 표시할수 있다. 따라서 가산기의 길이에는 관계없이 두단 문회로의 지연만이 생긴다.

긴 수들에 대해서는 이 수법이 지나치게 복잡해 진다. Nbit 가산기의 맨 웃자리비트에 대한 식을 평가해 보면 n-1 개 입구를 가진 하나의 OR 문회로와 2~n+1 개의 입구를 가진 n 개의 AND 문회로가 요구된다. 여기서 완전한 자리올림미리보기는 단번에 4~8bit 만을 진행하는것이 일반적이다. 그림 I-23 에는 4 개의 8bit 가산기로 32bit 가산기를 어떻게 구성하는가를 보여 주었다. 이런 경우에 자리올림은 4 개의 8bit 가산기를 거쳐 차례로 전달되여야 하지만 32 개의 1bit 가산기를 차례로 거치는것보다는 실질적으로 빨라 진다.

제 4 절. 순서회로

조합회로는 수자형콤퓨터의 기본적인 기능을 실현한다. 그러나 조합회로는 ROM 의 경우를 제외하고는 수자형콤퓨터의 연산에서 기본적인 요소인 정보의 기억 또는 보관을 하지 못한다. 이를 위하여 보다 복잡한 형태의 수자론리회로인 순서회로가 쓰인다. 순서회로의 현재출력은 현재입력에만 관계되는것이 아니라 입력신호의 지나간 결과에도 관계된다. 이것을 좀 다르게 그리고 보다 일반적인 쓸모 있는 방법으로 고찰하면 순서회로의 출력은 현재입력과 회로의 현재상태에 관계된다고 할수 있다.

이 절에서는 몇가지 간단하면서도 쓸모 있는 순서회로의 실례에 대해서 보게 된다. 순서회로는 조합회로를 써서 만든다.

1. 방아쇠회로

가장 간단한 형태의 순서회로는 방아쇠회로이다. 방아쇠회로에는 두가지 특성을 공유하는 여러가지 종류가 있다.

- 방아쇠는 쌍안정회로이다. 두 상태가운데서 한 상태에 존재하며 입력신호가 없으면 그 상태에 남아 있게 된다. 따라서 방아쇠는 1bit 기억기로 동작할수 있다.
- 방아쇠는 서로가 부정관계에 있는 두 출구를 가진다. 이 두 출구는 보통 Q 및 $\overline{Q}$ 로 표기된다.

S-R 빗장

그림 I-24 에는 S-R 방아쇠 또는 S-R 빗장으로 알려 진 공통적인 구성을 보여 주었다. 이 회로는 두개의 입구 S(설정)와 R(재설정), 두개의 출구 Q와 $\overline{Q}$ 그리고 귀환결

합을 엇걸어 놓은 NOR 문회로로 구성되여 있다.

우선 이 회로가 쌍안정회로라는것을 보기로 하자. S 와 R 가 둘다 0 이고 Q가 0 이라고 하자. 그러면 아래쪽 NOR 문회로의 입구는 Q=0 이고 S=0 이다. 따라서 출구 $\overline{Q}$=1 은 웃쪽 NOR 문회로의 입구가 $\overline{Q}$=1 및 R=0 으로 된다는것을 의미하며 출구는 Q=0 이 된다. 결국 회로의 상태는 내부적으로 변하지 않으며 S=R=0 인 동안은 오래동안 안정상태에 남아 있게 된다. 같은 리유로 역시 Q=1, $\overline{Q}$=0 인 상태도 R=S=0 에 대하여 안정상태이다.

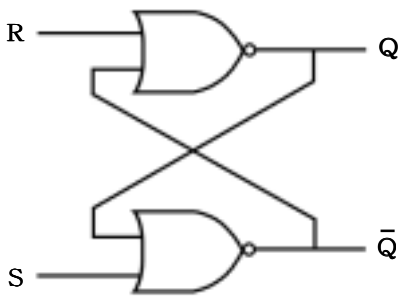



그림 I-24. NOR 문을 리용한 SR 빗장

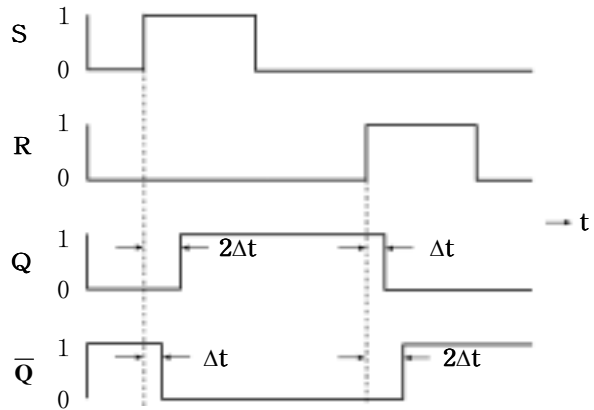



그림 I-25. NOR SR 빗장시간선도

이렇게 이 회로는 1bit 기억기로 동작한다. 출구 Q는 비트의《값》으로 볼수 있다. 입구 S 와 R 는 기억기에 각각 1 및 0 값을 쓰기 하는데 리용한다. 이것을 보기 위하여 Q=0, $\overline{Q}$=1,S=0,R=0 인 상태를 보자. 이제 S 가 값 1 로 변한다고 가정하자. 이때 아래쪽 NOR 에 대한 입구는 S=1, Q=0 이다. 약간한 시간지연 t 후에 아래쪽 NOR 문회로의 출구는 $\overline{Q}$=0 으로 된다(그림 I-25 를 참고). 따라서 이 순간에 웃쪽 NOR 문회로에 가해 지는 입구는 R=0, $\overline{Q}$=0 으로 된다. △ t 의 다른 문회로지연이 지나면 Q는 1 로 된다. 이것은 다른 하나의 안정상태이다. 이때 아래쪽 문회로의 입구는 S=1, Q=1 이고 출구 Q=0 을 유지하게 한다.

R 입구는 정반대의 기능을 수행한다. R 가 1 이 될 때 Q와 $\overline{Q}$의 이전상태에는 관계없이 Q=0, $\overline{Q}$=1 로 된다. 다시 2 △ t 의 시간지연이 일어 나고 이전의 안정상태가 다시 설정된다(그림 I-25).

S-R 빗장은 **특성표**라고 하는 진리값표와 비슷한 표에 의하여 정의할수 있다. 표는 현재상태와 입력의 함수로서 순서회로의 상태와 다음상태를 보여 준다. S-R 빗장인 경우에 그 상태는 Q의 값에 의하여 정의될수 있다. 표 I-10 ㄱ는 특성표를 얻는 과정을 보여 주고 있다. 표에서 입구 S=1, R=1 에 대해서는 허용하지 않는다. 그것은 이런 조합은 모순되는 출력(Q와 $\overline{Q}$가 둘다 0 과 같은)을 만들기때문이다. 이 표는 표 I-10 ㄴ와 같이 더 간단히 표시될수 있다. S-R 빗장의 동작과정의 해설은 표 I-10 ㄷ에서 보여 주었다.

동기식 S-R 방아쇠

S-R 빗장의 출구는 입구에서의 변화에 응답하여 짧은 지연시간후에 변한다. 이것은

표 I-10. S-R 빗장

ㄱ) 특성표

현재입구	현재상태	다음상태
SR	Q_n	Q_{n+1}
00	0	0
00	1	1
01	0	0
01	1	0
10	0	1
10	1	1
11	0	—
11	1	—

ㄴ) 간단화된 특성표

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	—

ㄷ)입구직렬에 대한 응답

t	0	1	2	3	4	5	6	7	8	9
S	1	0	0	0	0	0	0	0	1	0
R	0	0	0	1	0	0	1	0	0	0
Q_{n+1}	1	1	1	0	0	0	0	0	1	1

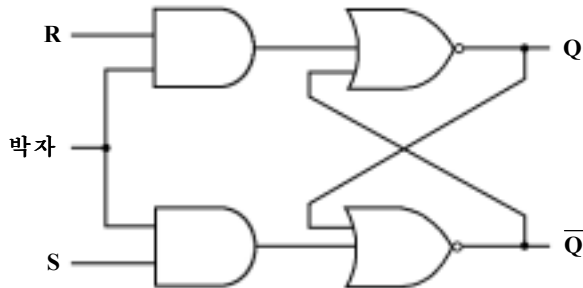



그림 I-26. S-R 방아쇠의 발진기

비동기동작으로 고찰된다. 수자형콤퓨터에서 가장 특징적인 점은 박자임풀스가 발생할 때에만 변화가 일어 나도록 사건들이 박자임풀스에 동기되여 있는것이다. 그림 I-26에는 이러한 구성을 보여 주었다. 이 회로는 동기식 S-R 방아쇠로 고찰된다. R 및 S 입력신호는 박자임풀스가 가해 지는 기간에만 NOR 문회로를 통과한다.

D 방아쇠

S-R 방아쇠에서 제기되는 한가지 문제는 R=1, S=1인 조건을 피해야 한다는것이다. 이를 위한 한가지 방도는 꼭 한개의 입구만 가지게 하는것이다. D 방아쇠는 이 문제를

완전히 풀어 준다. 그림 I-27에는 D방아쇠의 문회로에 의한 실현과 특성표를 보여 주었다. 반전기를 써서 박자신호입구가 아닌 두 AND 문회로에로의 입구가 서로 정반대가 되도록 한다.

D 방아쇠는 때때로 1bit 자료에 대한 효과적인 보관을 진행하므로 자료방아쇠라고도 한다. D 방아쇠의 출력은 언제나 입력에 가해 진 마지막값과 같게 된다. D 방아쇠는 또한 한 박자임풀스인 동안 그의 입구에 가해 진 0 또는 1 을 지연시키므로 지연방아쇠라고도 한다.

D	Q_{n+1}
0	0
1	1

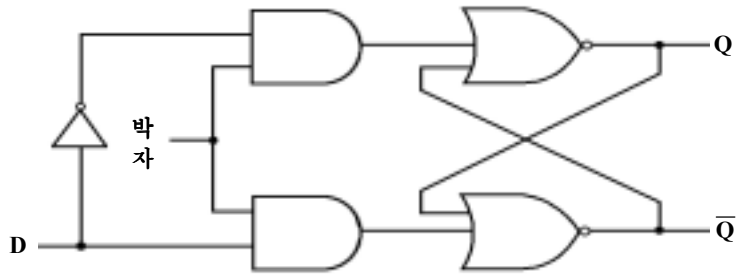



그림 I-27. D 방아쇠

J-K 방아쇠

또하나의 쓸모 있는 방아쇠는 J-K방아쇠이다. 이것은 S-R방아쇠와 같이 두개의 입구를 가지고 있다. 그러나 이 회로에서는 있을수 있는 모든 조합의 입력값이 다 허용된다. 그림 I-28 에는 J-K 방아쇠의 문회로에 의한 실현을 보여 주었다. 그림 I-29 에는 특성표를 보여 주었다(S-R 및 D방아쇠의 특성표와 함께). 첫 3가지 조합은 S-R방아쇠에 대한것과 같다. 입구가 없을 때도 안정하다. J 입구는 혼자서 출구를 1로 만드는 설정기능을 수행하며 K 입구는 혼자서 출구가 0이 되게 하는 지우기기능을 수행한다.

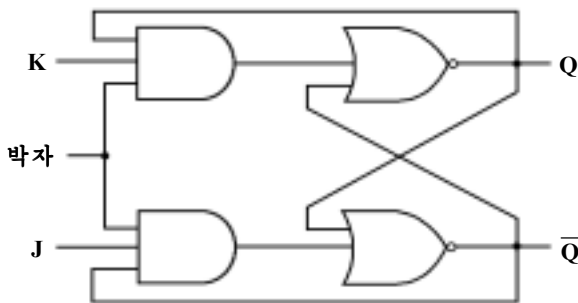



그림 I-28. J-K 방아쇠

J 와 K 의 두 입구가 1 일 때는 뒤집기기능이라고 하는 기능을 수행한다. 이때 출구는 반전한다. 그리하여 만일 Q가 1이고 J와 K에 1이 가해 지면 Q는 0이 된다. 독자들이 그림 I-28의 실현이 이 특성기능을 산생하는가를 검토해 보기 바란다.

2. 등록기

방아쇠의 한가지 응용실레로서 먼저 CPU 의 기본요소의 하나인 등록기에 대하여 보

기로 하자. 알고 있는바와 같이 등록기는 하나 또는 그이상의 2 진수자료를 기억하기 위하여 CPU 안에서 쓰이는 수자회로의 한가지이다. 등록기의 두가지 기본류형인 병렬등록기와 밀기등록기는 공통적으로 쓰인다.

병렬등록기

병렬등록기는 동시에 읽기 또는 쓰기를 할수 있는 1bit 기억기들의 묶음으로 되여있다. 이것은 자료를 보관하는데 쓰인다. 이 책에서 관통되여 론의해 온 등록기가 바로 병렬등록기이다.

그림 I-29. 기본방아쇠

그림 I-30 의 8bit 등록기는 병렬등록기의 동작을 보여 주고 있다. 여기서는 S-R 빗장들이 쓰인다. **입력자료따내기**라고 하는 조종신호는 D11~D18 의 신호선으로부터 등록기에로의 쓰기를 조종한다. 이 신호선들은 각이한 원천으로부터의 자료를 등록기에 적재할수 있도록 다중통로기의 출구로 된다. 출구도 비슷한 양식으로 조종된다. 특별한 특성으로는 등록기상태를 쉽게 0 으로 되게 하는 지우기신호선이 준비되여 있는것이다. 이것은 D 방아쇠로 구성된 등록기를 가지고는 쉽게 수행되지 않는다.

밀기등록기

밀기등록기는 직렬로 정보를 접수하거나 전송한다. 실례로 동기식 S-R 방아쇠로 구성된 5bit 밀기등록기를 보여 주는 그림 I-31 을 보자. 자료입구는 가장 왼쪽에 있는 방아

쇠에만 있다. 매개 박자임풀스와 함께 자료는 오른쪽으로 한 위치 옮겨 지며 가장 오른쪽의 방아쇠는 출구로 내보낸다.

밀기등록기는 직렬 I/O 장치에 대한 대면부로 리용될수 있다. 이외에도 밀기등록기는

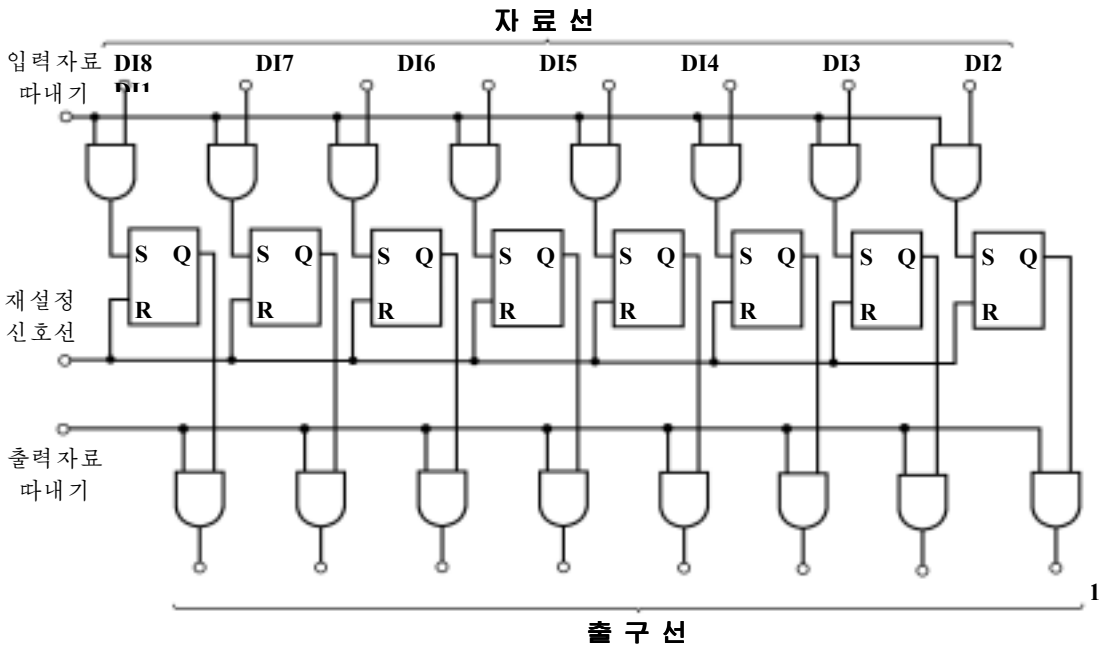



그림 I-30. 8bit 병렬등록기

ALU 안에서 론리밀기와 회전기능을 수행하는데 쓰인다. 이 두번째 능력에서 그것들은 직렬밀기등록기와 마찬가지로 병렬읽기/쓰기회로로 장비되여야 한다.

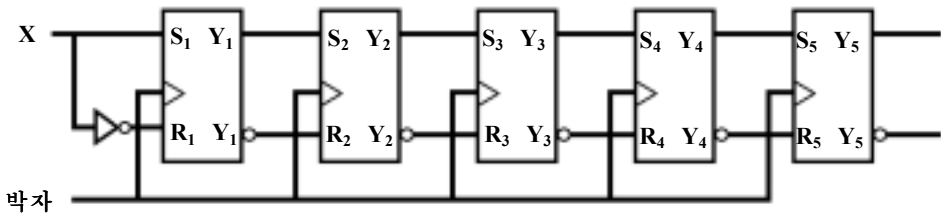



그림 I-31. 5 bit 밀기등록기

3. 계수기

순서회로의 또 하나의 쓸모 있는 부류는 계수기이다. 계수기는 그의 값이 쉽게 하나씩 증가되는 등록기이다. 이런 등록기는 n 개의 방아쇠로 만들어 지며 2^n-1 까지 계수할수 있다. 계수기가 그의 최대값을 넘어서 증가될 때에는 0 으로 설정된다. CPU 에서 계수기의 실례는 프로그람계수기이다.

계수기는 동작방식에 따라 비동기식 또는 동기식으로 설계될수 있다. 비동기식계수

기는 한 방아쇠의 출구가 다음방아쇠의 동작상태변화를 시동시키므로 상대적으로 속도가 뜨다. 동기식계수기에서는 모든 방아쇠가 같은 시각에 상태를 바꾼다. 동기식계수기는 속도가 훨씬 빠르므로 CPU 에서 많이 쓰이는 종류이다. 그러나 비동기식계수기는 서술로부터 론의를 시작하는것이 쓸모 있다.

비동기식계수기

비동기식계수기에서는 계수기를 하나 증가시키기 위해 일어 나는 변화가 한끝에서 시작하여 다른 끝까지 전달되므로 직렬계수기라고도 한다. 그림 I-32 에는 J-K 방아쇠를 리용한 4bit 계수기의 실현과 그의 동작과정을 보여 주는 시간선도를 함께 주었다. 시간선도는 신호가 방아쇠렬을 따라 가면서 움직일 때 생기는 전달지연을 보여 주지 않도록 리상화되였다. 가장 왼쪽에 있는 방아쇠의 출구(Q_0)는 맨 아래자리비트이다. 이 설계는 더 많은 방아쇠를 종속련결함으로써 얼마든지 긴 비트수를 가진 계수기로 확장할수 있다는것이 명백하다.

우의 실현에서 계수기는 매개 박자임풀스가 올 때마다 하나씩 증가된다. 매 방아쇠의 J와 K입구에는 늘 1이 설정된다. 이것은 박자임풀스가 올 때마다 Q의 출구는 반전된다는것을 의미한다. 상태변화는 박자임풀스의 내림면에서 일어 나도록 보여 주고 있다. 이런 방아쇠를 모서리시동방아쇠라고 한다. 임풀스자체가 아니라 박자임풀스에서의 상태이행에 따라 동작하는 방아쇠를 리용하면 복잡한 회로에서의 시간조종을 더 잘 할수 있다. 이 계수기에 대한 출구패턴을 조사하면 0000, 0001, …, 1110, 1111, 0000 과 같이 순환한다는것을 알수 있다.

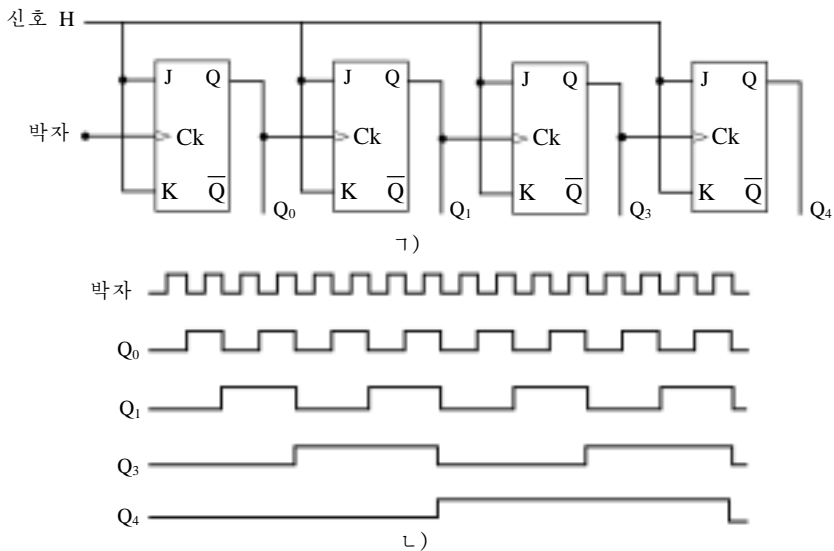



그림 I-32. 맥동계수기: ㄱ-순서회로, ㄴ-시간선도

동기식계수기

맥동계수기는 값이 변화될 때 지연이 동반되는 부족점을 가지고 있으며 이것은 계수기의 길이에 비례한다. 이런 부족점을 극복하기 위하여 CPU 는 동시에 계수기의 모든 방아쇠들의 상태가 변하게 되는 동기식계수기를 리용한다. 여기서는 3bit 동기식계수기에

대한 설계를 보기로 한다. 이를 위하여 먼저 동기식회로설계의 몇가지 기본개념을 설명한다.

3bit 계수기에는 3개의 방아쇠가 필요하다. J-K 방아쇠를 쓰기로 하자. 35개의 방아쇠의 반전되지 않은 출구들은 A, B, C 라고 하고 여기서 C 는 맨 아래비트가 되도록 하자. 첫 단계는 회로전반을 설계할수 있도록 J-K 입구와 출구의 관계를 반영하는 진리값표를 작성하는것이다. 그 진리값표를 그림 I-33 ㄱ에 보여 주었다. 여기서 왼쪽의 3개렬은 A, B, C 출구의 있을수 있는 조합을 보여 준다. 그것들은 계수기가 증가될 때 나타나

A	B	C	J_a	K_a	J_b	K_b	J_c	K_c
0	0	0	0	d	0	d	1	d
0	0	1	0	d	1	d	d	1
0	1	0	0	d	d	0	1	d
0	1	1	1	d	d	1	d	1
1	0	0	d	0	0	d	1	d
1	0	1	d	0	1	d	d	1
1	1	0	d	0	d	0	1	d
1	1	1	d	1	d	1	d	1

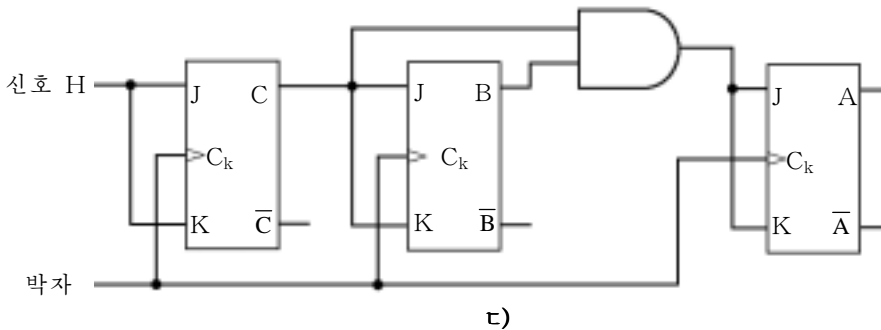
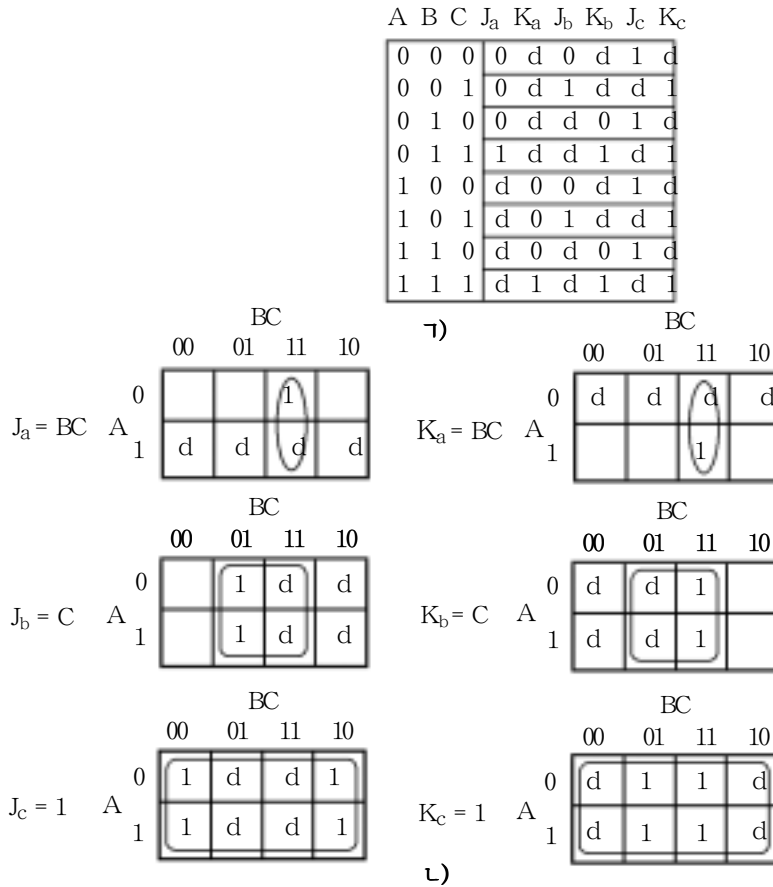



그림 I-33. 동기식계수기의 설계
ㄱ-진리값표, ㄴ-칸도표, ㄷ-론리도

는 순서대로 목록화되여 있다. 매개 행에는 A, B, C 의 현재값과 A, B, C 의 다음값을 얻는데 요구되는 3개의 방아쇠에 대한 입구들을 기입한다.

그림 I-33 ㄱ의 진리값표가 작성되는 방법을 리해하기 위하여 J-K 방아쇠에 대한 특성표를 고쳐 만드는것이 방조가 될수 있다. 그 표를 다시 상기하면 다음과 같다.

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\overline{Q}_{n+1}$

이 형식에서 표는 J 및 K 입구가 출구에 대하여 가지는 효과를 보여 준다. 이제 같은 정보에 대하여 다음과 같은 구조를 생각해 보자.

Q_n	J	K	Q_{n+1}
0	0	d	0
0	1	d	1
1	d	1	0
1	d	0	1

이 형식에서는 표가 입구와 현재의 출구가 알려 지면 다음출력값을 제공해 준다. 이것은 정확히 말하여 계수기 또는 임의의 순서회로의 설계에서 요구되는 정보이다. 이런 형식의 표를 려기표라고 한다.

그림 I-33 ㄱ로 되돌아 가서 첫행을 보자. 박자임풀스가 가해 지면 A 값이 0이고 B 값도 0상태에 있어 C 값이 0부터 1로 변한다. 려기표는 0의 출력상태를 유지하자면 입구가 J=0이고 K에 대해서는 무이금지조건이 되여야 한다는것을 보여 준다. 0부터 1에로의 이행이 일어 나자면 입구가 J=1 및 K=1 로 되여야 한다. 바로 이 값들을 표의 첫행에서 보여 주었다. 비슷한 리유로부터 표의 나머지부분들도 채울수 있다.

그림 I-33 ㄱ의 진리값표를 구성하면 A, B 및 C의 현재값의 함수로서 모든 J 및 K 입구들의 요구되는 값을 제시하는 표를 보게 된다. 그러면 칸도표의 도움으로 이 6 개의 함수에 대한 불식을 얻어 낼수 있다. 이에 대해서는 같은 그림의 ㄴ부분에서 보여 준다. 가령 J변수에 대한 칸도표(A출구를 만드는 방아쇠에 대한 J입구)는 식 J_a=BC라는 결과를 준다. 6개의 식이 유도되면 그것은 곧바로 그림의 ㄷ부분에 보여 준것과 같은 실제회로를 설계하는 자료로 된다.

련습문제

1. 다음불식에 대한 진리값표를 작성하시오.

ㄱ. $ABC+\overline{ABC}$　　ㄴ. $ABC+A\overline{BC}+\overline{ABC}$

ㄷ. $A(B\overline{C}+\overline{B}C)$　　ㄹ. $(A+B)(A+C)(\overline{A}+\overline{B})$

2. 다음론리식을 교환법칙에 따라 간단화하시오.

ㄱ. $A \bullet \overline{B} + \overline{B} \bullet A + C \bullet D \bullet E + \overline{C} \bullet D \bullet E + E \bullet \overline{C} \bullet D$

ㄴ. $A \bullet B + A \bullet C + B \bullet A$

ㄷ. $(L \bullet M \bullet N)(A \bullet B)(C \bullet D \bullet E)(M \bullet N \bullet L)$

ㄹ. $F \bullet (K + R) + S \bullet V + W \bullet \overline{X} + V \bullet S + \overline{X} \bullet W + (R + K) \bullet F$

3. 다음방정식에 데모르간의 정리를 적용해 보시오.

ㄱ. $F = \overline{V + A + L}$

ㄴ. $F = \overline{A} + \overline{B} + \overline{C} + \overline{D}$

4. 다음론리식을 간단화하시오.

ㄱ. $A = S \bullet T + V \bullet W + R \bullet S \bullet T$

ㄴ. $A = T \bullet U \bullet V \bullet W + X \bullet Y + Y$

ㄷ. $A = F \bullet (E + F + G)$

ㄹ. $A = (P \bullet Q + R + S \bullet T)T \bullet S$

ㅁ. $A = \overline{\overline{D} \bullet \overline{D} \bullet E}$

ㅂ. $A = Y \bullet (W + X + \overline{\overline{Y} + \overline{Z}}) \bullet Z$

ㅅ. $A = (B \bullet E + C + F) \bullet C$

5. 기본불연산인 AND, OR 및 NOR 로부터 XOR 연산을 구성하시오.

6. 3 입구 AND 기능을 수행할수 있는 론리도를 NOR 문회로와 NOT 문회로로 그리시오.

7. 4 입구 NAND 문회로에 대한 불식을 쓰시오.

8. 그림 I-34 에 보여 준것처럼 10 진수자의 7 쪼각표시기를 조종하는데 쓰이는 조합회로가 있다. 회로에는 4 개의 입구가 있다. 이 입구들은 조임형 10 진수표시에 쓰이는 4bit 부호를 제공한다(D_{10}=0000, …, 9_{10}=1001). 7 개의 출구는 주어 진 10 진수자를 표시하기 위하여 능동으로 되는 쪼각들을 규정한다. 일부 입구와 출구 조합은 필요 없다.

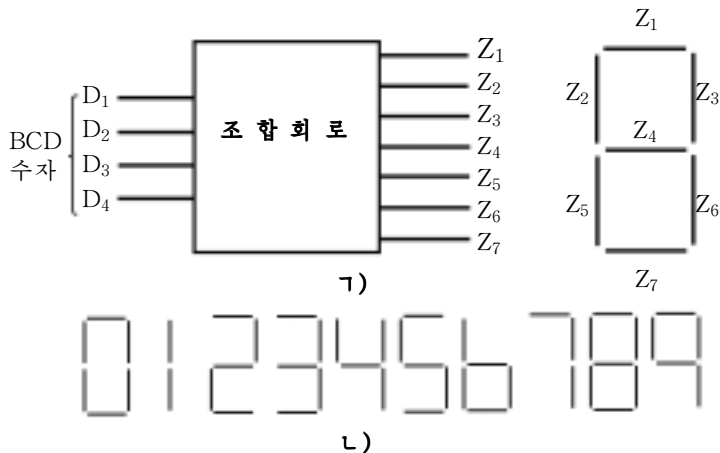



그림 I-34. 7 쪼각 LED 표시실례

ㄱ. 이 회로에 대한 진리값표를 만드시오.
ㄴ. SOP 형식의 진리값표를 표시하시오.
ㄷ. POS 형식의 진리값표를 표시하시오.

9. 8-1 다중통로기를 설계하시오.
10. 그림 I-15 가 분배기로 동작하도록 보충적인 신호선을 첨가하시오.
11. 그레이부호는 옹근수에 대한 2진수부호이다. 이 부호는 원래의 2진수표시와 차이난다. 이 부호에서는 련이어 있는 임의의 두수의 표시에서 1bit 만 차이나게 되여 있다. 이 부호는 순차적인 수가 발생되는 계수기나 상사-수자변환기와 같은 응용분야에 쓸모가 있다. 다만 한순간에 1bit 만이 바뀌기때문에 약간한 시간차에 의한 어떠한 모호성이 절대로 없다. 이 부호의 첫 8 개 요소는 다음과 같다.

2 진코드	그레이코드
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

2 진그레이부호변환회로를 설계하시오.

12. 4 개의 3 × 8 해신기(허가입구가 있는)와 2 × 4 해신기를 써서 5-32 해신기를 설계하시오.
13. 그림 I-22 의 옹근수가산기를 꼭 5 개의 문회로로 구성하시오(**암시:** 몇개의 문회로는 XOR 이다.).
14. 그림 I-22 를 고찰해 보자. 매개 문회로가 10ns 의 시간지연을 만들어 낸다고 하자. 이때 합출구는 30ns 후에 그리고 자리올림출구는 0ns 후에 안정된다. 32bit 가산기인 경우 총적인 가산시간은 얼마인가.

ㄱ. 그림 I-21 에서와 같이 자리올림미리보기기능이 없이 실현하시오.
ㄴ. 그림 I-23 에서와 같이 자리올림미리보기기능과 8bit 가산기를 리용하여 실현하시오.

부록 Ⅱ. 콤퓨터조직과 구성방식을 가르치기 위한 과제

많은 교원들은 연구 및 개발 과제가 콤퓨터조직과 구성방식의 개념을 명백히 리해하는데서 중요한 자리를 차지한다고 보고 있다. 이 지도계획이 없이는 학생들이 일부 기본개념과 구성요소들의 호상관계를 파악하기가 어려울수 있다. 이 지도계획의 목적은 이 책에서 언급된 개념을 보충하고 학생들에게 처리장치의 내부동작에 대한 보다 명백한 인식을 주며 학생들을 계발시키고 그들에게 학습내용을 완전히 파악시키자는데 있다.

이 책에서는 개념들을 될수록 명백히 리해시키고 공고화하기 위하여 200 개정도의 복습문제를 제시하였다. 많은 교원들은 계획에 이 교재를 보충하려고 한다. 이 부록에서는 몇개의 안내를 제공하고 있는데 여기에는 교원들을 위한 지도서에 쓰는 보충자료들을 서술하였다. 보충자료에는 3 가지 종류의 지도계획을 반영하고 있다.

- 연구개발지도계획
- 모의개발지도계획
- 참고문헌학습과제제시

1. 연구개발지도계획

강의에서 배운 기본개념들을 더잘 리해시키고 학생들의 연구능력을 키워 주는 효과적인 방도는 연구계획를 맡겨주는것이다. 이런 계획에는 판매제품의 Web 탐색을 비롯한 문헌탐색, 연구실험활동 및 표준화 등이 포함될수 있다. 개발계획은 개발팀이든가 계획이 작은 경우에는 개별적으로 줄수도 있다. 어떤 경우에는 주어야 할 계획의 제목과 수준이 적당한가를 평가하는것과 함께 학기시작전에 몇가지 계획에 대한 의견을 학생에게서 받는것이 가장 좋다. 연구계획을 위해 학생에게 줄 문건에는 다음과 같은것이 포함된다.

- 신청양식
- 최종보고양식
- 중간 및 최종제출날자가 있는 일정계획
- 가능한 계획목록

학생은 과제목록가운데서 하나를 선택하거나 자기에게 알맞는 계획을 제기할수 있다. 교원의 지도서에는 있을수 있는 연구제목목록뿐아니라 제안된 신청양식과 최종보고양식도 포함되여 있어야 한다.

2. 모의지도계획

처리장치의 내부조작에 대한 파악을 가지며 일부 설계해결책과 성능관계를 연구하고 평가하기 위한 우수한 방도는 처리장치의 기본요소들을 모의하는것이다. 이를 위하여 리용하고 있는 도구는 Simple Scalar 이다.

실제적인 장치실현과 비교해 볼 때 모의는 연구사업과 교육사업을 할때의 두가지 몇가지 우점이 있다.

- 모의를 하면 각이한 구성요소들의 변경과 그들의 성능특성의 변화, 그리고 그에 따르는 효과분석이 쉽다.
- 모의를 하면 또한 구체적인 성능에 대한 통계적자료를 얻을수 있으며 성능해결책에 대한 문제점들을 리해하기 쉽다.

SimpleScalar[BURG97]은 현대적인 처리장치와 체계의 어떤 계렬에 대한 실제프로그람을 모의하는데 쓰이는 도구들의 모임이다. 이 도구모임에는 번역프로그람, 기호번역프로그람, 련결편집프로그람 및 모의 및 현시도구가 포함되여 있다. SimpleScalar 는 매우 빠른 함수모의로부터 구체적인 비순차명령출구에 이르기까지를 포괄하는 처리장치모의와 비블로크화캐쉬와 투기실행을 지원하는 슈퍼스칼라처리장치모의를 제공한다. 명령모임구성방식과 구조적파라메터는 여러가지 실험에 따라서 변경할수 있다. 이 책에 대한 교원의 지도서에는 학생들이 SimpleScalar 를 어떻게 설치하고 기동시키는가에 대한 명령을 비롯한 일련의 명령들이 간결하게 서술되여 있다. 지도서에는 또한 일부 제안된 연구계획할당안이 들어 있다.

SimpleScalar 는 Windows NT 와 대부분의 UNIX 기반에서 실행될수 있는 휴대용프로그람제품이다. SimpleScalar 프로그람은 SimpleScalar Web 싸이트로부터 호출하여 설치할수도 있다. 그것은 비상업적용도에는 무료로 쓸수 있다.

3. 참고문헌학습과제

강의에서 나오는 개념을 보충하고 학생들에게 연구경험을 가지도록 하기 위한 또 한가지 좋은 방법은 읽고 분석한 참고문헌으로부터의 론문과제를 할당하는것이다. 교원지도서에는 우에 서술된 장마다 하나 혹은 두개의 론문제목들을 제시하였다. 론문은 인터네트나 대학도서관을 통하여 제공 받을수 있다. 지도서에는 또한 연구과제에 필요한 용어가 제시되였다.

용어해설

ㄱ

가상기억기 Virtual Storage
가상주소를 실지주소로 배치하는 콤퓨터체계의 사용자에 의하여 주소지정가능주기억기처럼 볼수 있는 기억기공간. 가상기억기의 크기는 주기억기위치의 실제적인 수에 의해서가 아니라 콤퓨터체계의 주소지정방식과 리용할수 있는 보조기억기의 크기에 의하여 제한된다.

간접주기 Indirect Cycle
CPU 가 간접주소를 직접주소로 변환하기 위하여 기억기접근을 진행하는 명령주기의 한 부분

간접주소 Indirect Address
주소를 가지는 기억기위치의 주소

고리형사슬 Daisy Chain
새치기자원들을 순서대로 련결하여 새치기우선권을 결정하기 위한 장치들의 호상련결방법

고밀도디스크 Compact Disk(CD)
수자화된 음성정보를 기억하는 지울수 없는 디스크

고점수표현체계 Fixed-Point Representation System
소수점이 합의된 일부 협약대로 배치된 수자렬에서 암시적으로 고정되는 소수점수체계

고체상태구성요소 Solid-State Component
조작이 고체에서의 전기적 혹은 자기적 현상의 조종에 의존하는 구성요소(e.g. 반도체 3극소자, 2극소자, 페리트자심)

국부변수 Local Variable
프로그람의 어느 규정된 한 부분에서만 정의되고 리용되는 변수

극소형콤퓨터 Microcomputer
처리장치가 극소형처리소자로 된 콤퓨터체계. 극소형콤퓨터는 극소형처리소자, 기억기, I/O 장치로 되여 있으며 이것들은 한개의 소편우에 있을수도 있고 그렇지 않을수도 있다.

극소형처리장치 Microprocessor
요소들이 한개 혹은 몇개의 집적회로로 극소화된 처리소자

금지된 새치기 Disabled Interrupt
CPU 를 조작하여 규정된 준위의 새치기요구를 무시하는 조건 또는 상태

기억기주소등록기 Memory Address Register(MAR)
처리장치에서 접근되는 기억주소를 가지는 등록기

기억기완충등록기 Memory Buffer Register(MBR)
기억기로부터 자료를 읽거나 기억기에 자료쓰기를 진행하는 등록기

기억기주기시간 Memory Cycle Time
기억기접근속도의 역수. 하나의 접근요구(읽기 혹은 쓰기)에 대한 응답과 다음접근요구에 대한 응답사이 최소시간이다.

기억주소식 I/O (Memory-Mapped I/O)
I/O 모듈과 외부장치를 기억주소를 리용하여 지정하는 방법. 하나의 주소공간이 주기억기와 I/O 주소지정에 다같이 리용되며 기억기와 I/O 읽기/쓰기에 같은 기계명령들이 리용된다.

기우성비트 Parity Bit
모든 수자들의 합이 늘 기수(기수성) 혹은 늘 우수(우수성)가 되도록 2 진수자들의 모임에 첨부되는 비트

기준주소 Base Address
콤퓨터프로그람의 집행에서 주소를 계산할 때 참조하는 수값

긴 명령단어 Very Long Instruction Word
다중조작을 포함하는 명령들의 리용을 말한다. 실지로 다중명령들은 단일한 단어에 포함된다. 대표적으로 VLIW는 콤파일러에 의하여 구성되는데 콤파일러는 같은 단어에 병렬로 실행될수 있는 조작을 배치한다.

관흐름 Pipeline
여러개의 명령을 동시에 실행할수 있도록 여러개의 단들로 이루어 진 처리장치조직

ㄷ

다중통로 Multiplexer Channel
여러개의 입출력장치들을 동시에 조작하도록 설계된 통로. 여러개의 I/O 장치가 자료의 항목들을 교차조작하면서 같은 시간에 레코드들을 전송할수 있다. 바이트다중선택통로, 블로크다중선택통로를 참고하시오.

다중처리장치 Multiprocessor
주기억기를 공동으로 접근할수 있는 여러개의 처리장치들을 가진 콤퓨터

다중프로그람방식 Multiprogramming
단일처리장치에 의해 둘 혹은 그이상의 콤퓨터프로그람의 집행을 교대로 진행하는 조작방식

다중화장치 Multiplexer
여러개의 입구를 단일출구에 련결하는 조합회로. 임의의 시간에 오직 하나의 입구만이 선택되여 출구로 나간다.

단일처리과정 Uniproccessing
처리장치들에서 명령들의 순차적인 처리과정 혹은 다중처리체계에서 처리장치들의 독립적인 리용

단항연산자 Unary Operator
1과 한 연산수만의 연산을 표시하는 연산자

동적 RAM Dynamic RAM
기억세포들이 축전기를 리용하여 실현된 자유기억기. 동적기억기는 주기적으로 재생하지 않으면 기억된 자료를 인차 잃어 버릴수 있다.

독립디스크들의 여분배렬 Redundant Array of Independent Disks (RAID)
물리적기억기용량의 한 부분을 나머지 부분에 기억된 사용자자료의 여유정보를 기억시키는데 리용하는 디스크배렬방식. 여유정보는 배렬의 성원디스크중의 하나 혹은 그에 대한 접근경로가 고장인 사건에서 사용자자료의 재생을 가능하게 한다.

등록기 Registers
CPU 내부의 고속기억기. 일부 등록기들은 사용자가 쓸수 있다. 즉 기계명령들을 통하여서만 프로그람작성자가 리용할수 있다. 기타 등록기들은 조종목적을 위하여 CPU에 의하여서만 리용된다.

동기식동기화(동기식시간일치) Synchronous Timing
모선에 대한 사건발생이 박자에 의해 결정되는 기술. 박자는 같은 너비의 시간틈을 가지며 사건들은 그 시간틈의 시작에서만 일어 난다.

디스케트 Diskette
보호함에 넣어 진 유연성자기디스크. 플로피디스크라고도 한다.

디스크묶음 Disk Pack
디스크구동기로부터 통채로 교체할수 있게 한 자기디스크의 조립묶음. 이것은 조작중에 조립묶음을 분리할수 있는 함에 넣어 진다.

디스크자료쪼각분산 Disk Stripping
론리적으로 련속인 자료블로크 혹은 쪼각들을 련속적인 배렬성원들에 대하여 회전순서로 배치하는 디스크배렬배치의 한 형태. 매 배렬성원들에 정확히 한 쪼각씩 배치하는 론리적으로 련속인 쪼각들의 묶음을 줄무늬라고 한다.

대칭다중처리 Symmetric Multi-proccessing(SMP)
조작체계가 임의의 필요한 처리소자 혹은 여러개의 차리소자들에서 동시에 처리할수 있는 다중처리의 한 형태

ㄹ

련상기억기 Associative Memory
기억위치가 그것들의 이름이나 주소에 의해서가 아니라 그 내용 혹은 그 내용의 일부분에 의하여 지적되는 기억기

류점수표현체계 Floating-Point Representation System
실수가 한쌍의 서로 구별되는 두수에 의하여 표시되는 수체계. 실수는 두수들중의 하나인 고점수부(결수부)와 두번째 수를 가리키는 류점수표현에서의 지수라고 하는 제곱에 대한 암시적인 류점수밑수에 의하여 얻어 진 값(지수부)의 적이다.

ㅁ

마이크로명령 MicroInstruction
기계명령보다 더 낮은 준위에서 처리장치안의 자료흐름과 처리순서를 조종하는 명령. 개별적인 기계명령들과 다른 기능들은 마이크로프로그람에 의해 수행된다.

마이크로조작 Micro-Operation
한박자의 임풀스내에 수행되는 CPU 의 개별적인 조작

마이크로프로그람 Microprogram
특별한 기억기안에 있는 마이크로명령의 렬로서 여러가지 기능을 수행하는데 동적으로 접근될수 있다.

마이크로프로그람화된 CPU Microprogrammed CPU
조종장치를 마이크로프로그람방식을 리용하여 실현한 중앙처리장치

마이크로프로그람언어 Microprogramming Language
마이크로프로그람을 서술하기 위하여 리용되는 명령모임

명령주소등록기 Instruction Address Register
실행하려는 다음명령의 주소를 보관하는데 리용되는 전용등록기

명령주기 Instruction Cycle
단일명령실행을 위해 CPU 에 의하여 수행되는 처리과정

명령형식 Instruction Format
비트렬로 된 콤퓨터명령의 지면배치. 명령형식은 명령의 구성요소에 따라서 여러개 마당으로 나눈다(례하면 조작코드, 연산수 등)

명령등록기 Instruction Register
해석을 위해 명령을 보존하는데 리용되는 등록기

모선 Bus
하나 또는 그 이상의 선들을 묶어서 구성한 공유전송경로. 일부 콤퓨터체계들에서 CPU, 기억기, I/O 등 구성요소들은 공통모선에 련결된다. 모든 구성요소들이 모선을 공유하므로 임의의 순간에는 오직 하나의 장치만이 모선을 리용할수 있다.

모선중재 Bus Arbitration
경쟁하는 모선관리자중에서 어느것이 모선에 대한 접근권을 가지는가를 결정하는 처리공정

모선관리기 Bus Master
모선에 대한 초기화 및 통신을 조종할수 있는 모선에 련결된 장치

모방 Emulation
주로는 하드웨어에 의하여 어떤 체계의 전체 혹은 일부를 모방하는것. 여기서 모방하는 체계는 같은 자료를 받아서 같은 프로그람을 실행하여 모방되는 체계와 같은 결과를 얻는다.

문회로 Gate
입구신호들에 대한 간단한 불연산으로 출구신호를 만드는 전기적회로

무조건이행 Unconditional Jump
그것을 규정한 명령이 어느때나 실행되도록 일어 나는 이행

밑수 Base
과학론문들에서 흔히 리용되는 수체계에서 표현되는 실수를 결정하기 위하여 지수부로 표시되는 제곱을 발생시키며 결수부에 곱해지는수(실례로 식 2.7 × $6.25^{1.5}$ = 42.1875 에서 수 6.25).

ㅂ

바이트 Byte
8개의 비트. Octet 라고도 한다.

바이트다중통로 Byte Multiplexer Channel
자료바이트들을 교차조작하는 다중통로. 블로크다중통로를 참고하시오. 선택기통로와 비교하시오.

반도체 Semiconductor
전기전도도가 부도체와 도체사이의 중간값을 가지는 규소나 게르마니움과 같은 고체결정물질. 3극소자와 고체상태요소를 만드는데 리용한다.

방아쇠 Flip-Flop
주어 진 순간에 두개의 안정상태중 하나의 안정상태를 가질수 있는 능동요소회로 혹은 장치, 쌍안정회로라고도 한다.

보조기억기 Secondary Memory
디스크와 테프를 비롯하여 콤퓨터체계밖에 위치한 기억기

부호-크기표현 Sign-Magnitude Representation
2진옹근수를 표시하기 위해 리용. N-bit 단어에서 맨아래자리비트는 부호(0=정, 1=부수)이고 남은 N-1 개의 비트들은 수의 크기를 표시한다.

분기예측 Branch Prediction
프로그람을 실행하기전에 분기결과를 예측하기 위하여 처리장치에서 리용하는 기구

분리된 입출력 Isolated I/O
I/O 모듈과 외부장치들의 주소지정하는 방법. I/O 주소공간은 주기억주소공간으로부터 개별적으로 취급된다. 특수한 I/O 기계명령들을 리용해야 한다. **기억기배치식 I/O** 와 비교하시오.

블로크다중통로 Block Multiplexer Channel
자료블로크를 교차조작하는 다중통로. **바이트다중통로**를 참고하시오.

벡토르 Vector
보통 순서화된 스칼라묶음에 의하여 특징 지어 지는 량

비동기식동기화(비동기식시간일치) Asynchronous Timing
모선에서 어떤 사건의 발생이 이전 사건의 발생에 뒤따르거나 의존하는 기술

비트 Bit
2진수체계에서 수자 0혹은 1중의 어느 하나를 나타낸다.
선택기통로와 비교하시오.

비균일기억기접근다중처리장치 Nonuniform Memory Access(NUMA) Multiprocessor
기억기에 있는 단어에 대한 주어 진 처리장치의 접근시간이 기억기단어의 위치에 따라 변하는 공유-기억기다중처리장치

비휘발성기억기 Nonvolatile Memory
그 기억내용이 전원을 꺼도 변하지 않는 기억기

ㅅ

사용자등록기 User-Visible Register
프로그람작성자가 참조할수 있는 CPU 등록기. 명령모임형식은 하나 혹은 그 이상의 등록기들이 연산수 혹은 연산수의 주소로 지정되는것을 허락한다.

산수-론리연산장치 Arithmetic and Logic Unit(ALU)
산수연산, 론리연산 그리고 그것들과 관계되는 다른 연산들을 수행하는 콤퓨

터의 한 부분

선택기통로 Selector Channel
한번에 하나의 I/O 장치만으로 동작하도록 설계된 I/O 통로기. 일단 I/O 장치가 선택되면 완료레코드가 한바이트씩 전송된다. **블로크다중통로**, **다중통로**를 참고

순서회로 Sequential Circiut
출구가 현재입구와 회로의 상태에 관계되는 수자론리회로. 따라서 순서회로는 기억기의 속성을 가진다.

슈퍼관흐름처리장치 Superpipelined Proccessor
명령관흐름이 많은 매우 작은 단들로 이루어 지도록 설계된 처리장치. 하나이상의 관흐름단을 한박자주기동안에 실행할수 있으며 많은 명령들이 같은 시간에 그 흐름선에 있을수 있다.

슈퍼스칼라처리장치 Superscalar Proccessor
다중명령관흐름을 포함하도록 설계된 처리장치. 다중명령관흐름을 포함하도록 설계된 처리장치. 하나이상의 명령을 동시에 같은 관흐름단에서 실행할수 있다.

실행주기 Execute Cycle
CPU 가 명령조작코드에 의해 규정된 연산을 수행하는 명령주기의 한 부분

새치기 Interrupt
외부적인 사건의 요구에 따라 현재 실행되고 있는 프로그람을 중지하고 외부적인 사건요구를 수행하기 위한 콤퓨터 처리의 한가지 방법

새치기주기 Interrupt Cycle
CPU 가 새치기를 검사하는 명령주기의 한부분. 만일 새치기가 가능하면 CPU 는 현재프로그람상태를 보존하고 새치기-조종기루틴에서 그 처리를 계속하게 한다.

새치기구동식 I/O Interrupt-Driven I/O
I/O 의 한 형태. CPU 는 I/O 지령을 현시하여 련속명령실행을 계속하며 련속명령실행이 완료되였을 때 I/O 모듈에 의하여 중단된다.

ㅈ

자기원판 Magnetic Disk
자료를 기억할수 있는 단면 혹은 량면우에 자기적인 표면층을 가진 두꺼운 원판

자기테프 Magnetic Tape
자기기록에 의해 자료를 기억할수 있는 자기적인 표면층을 가진 테프

자동첨수화방식 Autoindexing
첨수등록기가 매 기억기참조에서 자동적으로 증가하거나 감소하는 첨수주소화방식

자료모선 Data Bus
자료전송을 위하여 리용되는 체계모선의 한 부분

자료통신 Data Communication
장치들사이에서의 자료전송. 이 말은 주로 I/O 장치인 경우에 쓰인다.

자유접근기억기 Random-Access Memory (RAM)
매 주소지정가능위치가 동일한 주소지정기구를 가지는 기억기. 주어 진 위치를 접근하는 시간은 선행접근순서와 무관계하다.

전역변수 Global Variable
콤퓨터프로그람의 한 부분에서 정의되여 다른 부분에서도 리용할수 있는 변수

절대주소 Absolute Address
어떤 중간참조가 없이 기억위치나 장치를 지적하는 콤퓨터언어에서의 주소

정적 RAM Static RAM
기억세포가 방아쇠를 리용하여 실현된 RAM. 정적 RAM 은 전원이 공급된 조건에서만 자료를 유지한다. 주기적인 재생이 필요 없다.

조건코드 Condition Code
다음연산결과에 영향을 주는 코드.

CPU 는 하나 혹은 여러개의 조건코드를 가질수 있다. CPU 내부에 따로 기억시킬수도 있고 보다 큰 조종등록기에 기억시킬수도 있다.

조종모선 Control Bus
조종신호들의 전송을 위하여 리용되는 체계모선의 한 부분

조종등록기 Control Registers
CPU 의 동작을 위하여 리용되는 CPU 등록기. 이 등록기들의 대부분은 사용자가 볼수 있다.

조종기억기 Control Storage
마이크로코드를 포함한 기억기의 한 부분

조종장치 Control Unit
ALU 연산, CPU 내부에서의 자료옮김, 외부대면부(례하면 체계모선)를 통하여 자료와 조종신호의 교환을 비롯한 CPU 의 조작을 조종하는 CPU 의 한 부분

조합회로 Combinational Circuit
어떤 주어 진 순간의 출력값들이 그때의 입력값에만 의존하는 론리회로. 조합회로는 기억가능성이 없는 순서회로의 특수경우이다.

조건이행 Conditional Jump
그것을 규정하는 명령이 실행되거나 규정된 조건이 만족될 때에만 일어 나는 이행. 무조건이행과 반대의미를 가진다.

조작코드 OPCode
조작코드(OPeration Code)를 나타내는 준말

조작체계 Operating System
프로그람의 집행을 조종하며 자원할당, 순서짜기, 입출력조종, 자료관리와 같은 봉사를 제공하는 쏘프트웨어

조작코드 Operation Code
콤퓨터의 연산을 표시하는데 리용되는 코드. 보통 OP 코드로 간략화한다.

주기억기 Main Memory
명령들과 다른 자료를 순차실행 혹은 처리를 위하여 등록기에 직접 넣을수 있는 프로그람-주소지정가능기억기

주소모선 Address Bus
주소전송을 위해 리용되는 일부 체계모선. 일반적으로 주기억기주소나 I/O 장치를 지적한다.

주소공간 Address Space
참조할수 있는 주소(기억기, I/O)의 범위이다.

주변장치 Peripheral Equipment(IBM)
콤퓨터체계에서 적당한 처리장치가 외부와 통신을 할수 있게 하는 임의의 장치

중앙처리장치 Central Processing Unit(CPU)
명령을 꺼내여 실행하는 콤퓨터의 한 부분. 이것은 산수-론리연산장치, 조종장치, 등록기들로 구성되여 있다. 간단히 처리소자라고 한다.

지우기가능한 빛디스크 Erasable Optical Disk
광학기술을 리용한 디스크로서 쉽게 지우기나 재쓰기할수 있는 디스크. 3.25 인치와 5.25 인치디스크들이 리용된다. 대표적인 용량은 650MB 이다.

직교성 Orthogonality
두 변수 혹은 두 차원이 서로 독립인 성질. 이 말은 명령모임의 전후관계에서 일반적으로 명령의 다른 요소(주소방식, 연산수개수, 연산수길이)들이 조작코드와 독립(조작코드에 의하여 결정되지 않음)라는것을 가리키는데 리용된다.

직접값주소 Immediate Address
주소가 아니라 연산수의 값을 가지는 주소부의 내용. 이것을 **령-준위주소**라고도 한다.

직접접근 Direct Access
자료의 물리적위치를 지적하는 주소들을 리용하여 상대적위치의 기억기에 차례로 자료를 기억시키거나 자료를 불러낼수 있는 능력

직접주소 Direct Address
연산수로 취급되는 자료가 들어 있는 기억위치를 지적하는 주소

직접기억기접근 Direct Memory Access(DMA)
DMA 모듈이라고 불리우는 특수한 모듈이 주기억기와 I/O 모듈사이의 자료교환을 조종하는 입출력의 한 형식 CPU 는 DMA 모듈에 자료블로크의 전송을 위한 요구신호를 보내며 전체 블로크가 전송된후 새치기가 발생한다.

진리값표 Truth Table
모든 가능한 입력값들의 조합을 목록화하며 매 조합에 대하여 그 출력값을 지적하여 론리함수를 서술하는 표

집적회로 Integrated Circuit(IC)
전자구성요소들과 그것들이 호상 련결을 통합하여 놓은 규소와 같은 고체물질의 소편

ㅊ

참조의 국부성 Locality of Reference
짧은시간주기에서 반복적으로 같은 묶음의 기억기위치들을 접근하는 처리장치의 경향

첨수주소 Indexed Address
콤퓨터명령의 실행전 혹은 실행동안에 첨수등록기내용에 의하여 변경되는 주소

첨수주소화 Indexing
첨수등록기를 리용하여 주소를 변경하는 기술

첨수등록기 Index Register
명령이 실행되는 동안 그 내용이 연산수주소를 변경시키는데 리용될수 있는 등록기. 이 등록기는 계수기로 리용할수도 있다. 첨수등록기는 순환고리의 실행을 조종하거나 표검색을 위하여 스위치로서 혹은 지적자로서 배렬리용을 조종하는데 쓸수 있다.

처리장치 Processor
콤퓨터에서 명령을 해석하고 실행하는 기능장치. 처리장치는 적어도 명령조종장치와 산수연산장치로 구성된다.

처리장치주기시간 Processor Cycle Time
가장 짧게 잘 정의된 CPU 마이크로조작에 요구되는 시간. 이것은 모든 CPU 작용을 재기 위한 기초시간단위이다. **기계주기시간**이라고도 한다.

축적기 Accumulator
단일주소명령형식에서 CPU 등록기의 이름. 축적기 혹은 AC 는 암시적으로 명령에서 두 연산자중 어느 하나로 된다.

체계모선 System Bus
주요콤퓨터구성요소(CPU, 기억기, I/O)들을 호상 련결하는데 리용되는 모선

ㅋ

콤퓨터명령 Computer Instruction
설계된 콤퓨터의 처리장치가 인식할수 있는 명령. 기계명령과 같은 말이다.

콤퓨터명령모임 Computer Instruction Set
명령연산수에 부여되여 있는 여러가지 의미를 서술하는 콤퓨터명령모임. 기계명령모임과 같은 말이다.

클라스터(병렬 혹은 집합콤퓨터) Cluster
하나의 콤퓨터처럼 느낄수 있게 단일한 계산자원에 대하여 함께 작업하는 호상 련결된 여러개의 콤퓨터들의 묶음

캐쉬일치성규약 Cache Coherence Protocol
모든 자료접근이 늘 주기억기단어의 내용중에서 가장 최근의것을 요구하도록 여러개의 캐쉬중에 자료유효성을 보존하는 기구

캐쉬행 Cache Line
캐쉬꼬리표와 관련되는 자료블로크, 캐쉬와 기억기사이의 전송단위

캐쉬기억기 Cache Memory
특수한 완충기억기. 이것은 처리장치에 의하여 다음에 요구하거나 주기억기로부터 자동적으로 얻게 되는 주기억기안의 자료나 명령을 복사, 보관, 유지하는데 리용되는 주기억기보다 더 작고 더 빠른 완충기억기이다.

ㅌ

탄창 Stack
복귀되는 다음항이 제일 마지막에 기억된 항이 되도록 구성하고 유지하는 목록. 후입선출목록(LIFO)이라고도 한다.

투기실행 Speculative Execution
분기중 어느 한 경로에 따르는 명령들의 실행. 이 분기가 일어 나지 않았다는것이 후에 판명되면 투기실행결과는 무시된다.

ㅍ

펌웨어 Firmware
ROM에 기억된 마이크로코드

프로그람계수기 Program Counter
명령주소등록기

프로그람식론리배렬 Programmable Logic Array(PLA)
그것들의 호상련결이 특별한 론리함수기능을 수행하는 프로그람일수 있는 문회로들의 배렬

프로그람식읽기전용기억기 Programmable Read-Only Memory(PROM)
내용을 오직 한번만 설정할수 있는 반도체기억기. 쓰기처리는 전기적으로 수행되며 사용자는 초기소편을 만든후에 한번만 쓰기를 할수 있다.

프로그람식 I/O Programmed I/O
CPU가 I/O모듈에 I/O지령을 출력하고 그다음 그 조작이 진행되기전에 완료되기를 대기해야 하는 입출력의 한 형식

프로그람상태단어 Program Status Word(PSW)
명령이 실행되는 순서를 지적하며 콤퓨터체계의 상태를 보존하고 지적하는데 리용되는 기억기령역. **처리장치상태단어**라고도 한다.

프로쎄스 Process
집행중의 프로그람. 프로쎄스는 조작체계에 의해 조종되며 순서가 결정된다.

프로쎄스조종블로크 Process Control Block
조작체계에서 프로쎄스의 상태표시. 이것은 프로쎄스의 특성과 상태에 대한 정보를 포함하고 있는 자료구조이다.

페지 Page
가상기억기체계에서 가상주소를 가지고 실지기억기와 보조기억기사이 장치로 전송되는 고정길이블로크.

페지오유 Page Fault
참조단어를 가지는 페지가 주기억기에 없는 경우 발생한다. 이것은 새치기를 발생시켜 조작체계가 요구하는 페지로 가져 올것을 요구한다.

페지틀 Page Frame
페지를 보관하는데 리용되는 주기억기의 령역

ㅎ

허용된 새치기 Enabled Interrupt
CPU를 조작하여 규정된 준위의 새치기요구를 허락하는 조건 또는 상태

해신기 Decoder
몇개의 입구신호와 여러개의 출구신호선을 가진 소자. 출구와 입구 신호조합사이에는 1대 1 대응관계가 있다.

핵심 Nucleus
기본적이며 가장 자주 리용되는 기능을 가진 조작체계의 부분. 보통 핵심은 주기억기에 상주한다.

휘발성기억기 Volatile Memory
기억기내용을 유지하기 위해 직류전원이 요구되는 기억기. 전원이 차단되면 기억된 정보를 잃어 버린다.

ㄲ

꺼내기주기 Fetch Cycle
CPU가 실행하려고 하는 명령을 기억기로부터 꺼내는 명령주기의 한 부분

ㅇ

아셈블리언어(기호언어) Assembly Language
일반적으로 그 명령들이 콤퓨터명령과 1:1 로 대응되며 마크로명령과 같은 명령을 리용할수 있는 콤퓨터지향언어. **콤퓨터-의존언어**라고도 한다.

완충기 Buffer
한 장치에서 다른 장치에로 자료를 전송할 때 자료흐름의 속도나 사건발생시간의 차이를 맞추기 위하여 리용되는 기억기

연산수 Operand
연산을 수행하는 기능실체

요구페지화 Demand Paging
필요한 순간에 보조기억기로부터 실지기억기에로 페지전송

오유수정코드 Error-Correcting Code
매개 문자 혹은 신호를 규정된 코드구성규칙에 맞춘 코드. 이 규칙을 리용하여 오유의 검출과 검출된 오유의 일부 혹은 전부를 자동적으로 수정한다.

오유검사코드 Error-Detecting Code
매개 문자 혹은 신호를 규정된 코드구성규칙에 맞춘 코드. 이 규칙을 리용하여 오유검출을 진행한다.

일반등록기 General-Purpose Register
등록기묶음안에서 늘 명시적으로 주소화할수 있는 등록기. 이것은 여러가지 목적에 쓸수 있다(례를 들면 축적기, 첨수등록기, 자료에 대한 특수한 조종자로 쓸수 있다).

입출력 Input-output(I/O)
입력이나 출력 혹은 둘다를 의미. 콤퓨터와 그에 직접 련결된 주변장치사이에서 자료의 입출력을 의미하는 경우가 많다.

읽기전용기억기 Read-Only Memory(ROM)
그 내용을 기억기가 파괴되지 않는 한 교체할수 없는 기억기. 지울수 없는 기억기

예측실행 Predicated Execution
개별명령들의 조건실행을 지원하는 기구. 이것은 분기명령의 두 갈래에 대한 실행이 투기적으로 가능하게 하며 결국 취해 진 분기결과를 보존하게 한다.

＊＊＊

1 의 보수표현 Ones Complement Representation
2 진수등록기들을 표시하는데 리용된다. 정의 옹근수는 부호-크기표현법으로 표시된다. 부의 옹근수는 같은 크기의 정의 옹근수표현에서 매 비트를 반전시켜 표시한다.

2 의 보수표현 Twos Complement Representation
2 진옹근수를 표시하는데 리용. 정의 옹근수는 부호-크기표현법으로 표시된다. 부수는 그 수의 1 의 보수표시에 1 을 더하여 표시한다.

2 **진연산자** Binary Operator
두개의 연산수에 대한 연산을 표현하는 연산자

＊＊＊

ASCII
정보교환을 위한 미국표준코드. ASCII 는 수자, 영문자, 특수한 인쇄문자들에 리용되는 7bit 코드이다. 또한 인쇄나 표시는 되지 않지만 일부 특수한 조종기능을 수행하는 조종문자들에 대한 코드들도 포함한다.

CD-ROM
읽기만을 할수 있는 빛소형디스크를 말한다. 콤퓨터자료를 기억시키기 위하여 리용되는 지울수 없는 디스크이다. 표준체계는 12Cm 원판으로 되여 있으며 550Mbyte 이상을 기억시킬수 있다.

G:10 억을 의미하는 앞붙이

I/O 통로 I/O Channel

섬세한 I/O 조작에서 CPU 의 부담을 덜어 주는 상대적으로 복잡한 I/O 모듈. I/O 통로는 CPU 참가를 요구하지 않고 주기억기로부터 I/O 지령렬을 실행한다.

I/O 모듈 I/O Module

콤퓨터의 주요구성부분의 하나. 하나 혹은 그이상의 외부장치(주변장치)의 조종과 그 장치들과 주기억기 혹은 CPU 등록기들사이 자료교환에 리용할수 있다.

I/O 조종기 I/O Controller

CPU 나 I/O 통로로부터 세부적인 조종을 요구하는 상대적으로 단순한 I/O 모듈. **장치조종기**라고도 한다.

I/O 처리장치 I/O Processor

그자체의 전용 I/O 명령 혹은 일부 경우에 일반등록기의 기계명령을 실행할 수 있는 자체처리장치를 가진 I/O 모듈

K: 2^{10}=1024 인 앞붙이. 따라서 2Kb=2046bit 이다.

M: 2^{20}=1048576 을 의미하는 앞붙이. 따라서 2Kb=2_220bit 이다.

WORM

쓰기-한번 읽기-여러번을 의미한다.

한번-복사(single-copy)디스크를 상품화함으로써 CD-ROM 보다 더 쉽게 쓰기할수 있게 된 디스크.쓰기조작이 한번 수행된후에는 CD-ROM 과 같이 읽기만 할수 있다. 가장 일반적인 크기는 5.25inch 로서 자료를 200～800Mbyte 보존할수 있다.

참고문헌

ALEX93 Alexandridis, N. *Design of Microprocessor-Based Sytems*. Englewood Cliffs, Nj: Prentice Hall, 1993.

SHAN95 Shanley,T., and Anderson, D. *PCI Systems Architecture*. Richardson, TX: Mindshare Press, 1995.

SOLA94 Solari, E,, and Willse, G. *PCI hardware and software*: Architecture and Desing. San Diego, CA: Annabooks,1994.

ADAM91 Adamek, J. *Foundations of coding*.New York: Wiley,1991

AGAR89 Agarwal, *Analysis of Cache Performance for Operating Systems and Multi-programming.* Boston: Kluwer Academic Publishers, 1989.

ANDE98 Adnerson, D., and Shanley, T.*Pentium Pro and Pentium II System Architecture*. Reading, MA: Addison-Wesley,1998.

BLAH83 Blahut, R.*Theory and Practice of Error Control Codes*. Reading, MA: Addison-Wesley, 1983.

HAND98 Handy, J. The *Cache Memory* book.san Diego: Academic Press, 1993.

HIGB90 Higbie, L. ”Quick and Easy Cache Performance Analysis. ”*Computer Architecture News*, June 1990.

MCEL85 McEiece, R. ”The Reliability of Computer Memories. ”*Scientific American*, January 1985.

MOTO97 Mororola, Inc. *PowerPC Microprocessor Family: The Programming Environments for 32-bit Microprocessors*. Denver, CO, 1997

PRIN91 Prince, B. *Semiconductor Memories.* New York: Wiley, 1991

PRIN96 Prince, B. *High Performance Menories: New Architecture DRAMs and SRAMs, Evolution and Function* .New York: Wiley, 1996

SHAN95 Shanley. T. *PowerPC: System Architecture. Randing*, MA: Addison-Wesley, 1995.

SHAR97 Sharma, A. *Semiconductor memories:Technology,Testing,and Reliability*. New York: IEEE Press, 1997.

SMIT82 Smith, A.”Cache Memories.”*ACM Computing Sureys*, September 1992.

GOLD91 Goldberg, D. “What Every Computer Scientist Should Know About Floationg –Point Arithmetic.” *ACM Computing Surveys*, March 1991. Available at http://www.validgh.com/

KNUT98 Knuth,D. *The Art of Computer Programming*, *Volume 2: Seminumerical Algorithms*. Reading, MA:Addison-Wesley, 1998

MORG92 Morgan, D. *Numerical Methods*. San Mateo, CA: M&T Books, 1992

OBER97a Oberman,S., and Flynn, M. ”Design Issues in Division and Other Floating-Point Operations.” *IEEE Transactions on Computers*, February 1997

OBER97b Oberman, S., and Flynn, M. “Division Algorithms and Implementations.” *IEEE Transactions on Computers*, August 1997

OMON94 Omondi,A. *Computer Arithmetic Systems:Algorithms, Architecture and Implementations*. Englewood cliffs, NJ: Prentice Hall, 1994

SODE96 Soderquist,P., and Leeser,M."Area and Perfomance Tradeoffs in Floating-Point Divide and Square-Root Implementations." *ACM Computing Surveys*, September 1996

SWAR90 Swartzlander,E., editor, *Computer Arithmetic*, Volumes ▪ and ▪ . Los Alamitos, CA: IEEE Computer Society Press, 1990

WALL90 Wallis,P. *Improving Floating-Point Programming*. New York: Wiley, 1992

BREY97 Brey, B. *The Intel Microprocessors: 8086/8066, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Processor.* Upper Saddler River, NJ: Prentice Hall, 1997.

DEWA90 Dewar, R., and Smosna, M. *Microprocessors: A Programmer's View.* New York: McGraw-Hill, 1990.

HAYE88 Hayes, J. *Computer Architecture and Organization, 2nd edition.* New York: McGraw-Hill, 1988.

HENN96 Hennessy, J., and Patterson, D. *Computer Architecture: A Quantitative Approach.* San Mateo, CA: Morgan Kaufmann, 1996.

IBM94 International Business Machines, Inc. *The PowerPC Architecture : A Specification for a New Family of RISC Processors.* San Francisco, CA: Morgan Kaufmann, 1994.

TANE90 Tanenbaum, A. *Structured Computer Organization.* Englewood Cliffs, NJ: Prentice Hall, 1990.

WEIS94 Weiss, S., and Smith, J. *POWER and PowerPC.* San Francisco: Morgan Kaufmann, 1994.

BLAA97 Blaauw, G., and Brooks, F. *Computer Architecture: Concepts and Evolution.* Reading, MA: Addison-Wesley, 1997

FLYN85 Flynn, M.: Johnson, J.: and Wakefield, S. "On Instruction Sets and Their Formats." *IEEE Transactions on computers*, March 1985.

BREY97 Brey, B. *The Intel Microprocessors: 8086/8066, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Processor.* Upper Saddle River, NJ:Prentice Hall,1997.

CRAG92 Cragon, H. Branch Strategy Taxonomy and Performance Models. Los Alamitos, CA:IEEE Computer Society Press, 1992.

DUBE21 Dubey,P. and Flynn,M. "*Branch Strategies: Modeling and Optimization.*" *IEEE Transactions on Computers*, October 1991.

BASH91 Bashteen, A.; Lu .; and Mullan,J." A Superpipeline Approach to the MIPS Architecture." *Proceedings, COMPCON Spring '91*, February 1991.

DEWA90 Dewar, R. , and Smosna, M. *Microprocessors: A Programmer's View.* New York: McGraw-Hill, 1990

HENN96 Hennessy, J., and Patterson, D. *Computer Architecture: A Quantitative Approach.* San Mateo, CA: Morgan Kaufmann, 1996.

KANE92 Kane,G., and Heinrich, J. *MIPS RISC Architecture.* Englewood Cliffs, NJ: Prentice Hall, 1992

MIRA92 Mirapuri, S.; Woodacre,M.; and Vassghi,N. “The MIPS R4000 Processor.” *IEEE Micro,* April 1992.

PATT98 Patterson, D., and Hennessy, J. *Computer Organization and Design: The Hardware/Software Interface.* San Mateo,CA: Morgan Kaufmann, 1998.

WARD90 Ward, S.,and Halstead, R. *Computation Structures*. Cambridge, MA: MIT Press, 1990.

MANO97 Mano ,M. Logic and Design Fundermen. Upper Saddle River,NJ:Prentice Hall,1997

AWRD90 Ward,S.,and Halstead,R. Computation Structures. Cambridg,MA:MIT Press, 1990

DULO98 Dulong, C.《The IA-64 Architecture at Work.》 *Computer*, July 1998.

HWU98 Hwu, W.《Introduction to Predicated Execution》, *Computer* , January 1998.

JOHN91 Johnson, M. *Superscalar Microprocessor Design*. Englewood Cliffs, NJ: Prwntice Hall, 1991.

JOUP89a Jphnson, N., and Wall, D.《Available Instruction-Level Parallelism for Super-scalar and Superpipelined Machines.》 *Proceedings, Third International Conference on Architectural Support for Programming Languages and Operating Systems,* April 1989.

KUGA91 Kuga, M.; Murakami,K.; and Tomita, s.《DSNS(Dynamically-Hazard Resolved, Statically-Code-Scheduled, Nonuniform Superscalar):Yet Another Superscalar Processor Architecture.》 *Computer Architecture News*, June 1991.

LEE91 Lee, R.; Kwok, A.; and Briggs, F. 《The Floating Point Performance of a Superscalar SPARC Processor.》 *Proceedings, Third International Conference on Architectural Support for Programming Languages and Operating Systems*, Apil 1989.

NORM98 Normoyle, K., et al.《UltraSPARC-▪ i: Expanding the Boundaries of a Statem on a Chip.》*IEEE Micro*,March/April 1998.

PAPW96 Papworth, D.《Tuning the Pentium Pro Microarchitecture.》*IEEE Micro*, Apil 1996

POPE91 Popescu, V., et al.《The Metaflow Architecture.》 *IEEE Micro*, June 1991.

POTT94 Potter,T., et al.《Resolution of Data and Control-Flow Dependencies in the PowerPC 601.》*IEEE Micro*,October 1994.

SHAN95 Shanley, T. PowerPC *System Architecture*. Reading, MA: Addison-Wesley, 1995.

SHAN98 Shanley, T. Pentium Pro and Pentium *System Architecture*. Reading, MA: Addison-Wesley, 1998.

SIMA97 Sima, D.《Superscalar Instruction Issue.》*IEEE Micro*, September/ October 1997.

SMIT95 Smith, J., and Sohi, G.《The Microarchitecture of Superscalar Processors.》 *Proceedings of the IEEE,*December 1995.

WALL91 Wall,D.《Limits of Instruction-Level Parallelism.》 *Proceedings, Fourth International Conference on Architectural Support for Programming Languages and Operating Systems,* Apil 1991.

YEAG96 Yeager,K.《The MIPS R 10000 Superscalar Microprocessor.》*IEEE Micro*, Apil 1996.

CHEN94 Chen, p.;Lee, E.; Gibson, G.; Katz, R.;and Patterson, D.”RAID:High Performance, Reliable Secondary Storage.” *ACM Computing Surveys*, June 1994.

FRIE96 Friedman, M. “RAID Keeps Going and Going and…” *IEEE Spectrum*, April 1996.

MANS97 Mansuripur, M.,and Sincerbox, G.”Principles and Techniques of Optical Data Storage.” *Proceedings of the IEEE*, November 1997.

MARC90 Marchant, A. *Optical Recording.* Reading, MA:Addison-Wesley, 1990.

MASS97 Massiglia, P. *The RAID Book: A Storage System Technology Handbook*. St. Peter, MN:The Raid Advisory Board, 1997.

MEE96a Aee, C., and Daniel, E., eds. *Magnetic Recording Technology*. New York: McGraw-Hill, 1996.

MEE96b Mee, C.,and Damiel,E.,eds. *Magnetic Storage Handbook.* New York: McGraw-Hill, 1966.

ROSC97 Rosch, W. Winn L. Rosch *Hardware Bible*. Indianaolis, IN: Sams, 1997.

CART96 Carter,J.*Microprocesser Architecture and Microprogramming.* Upper Saddle River, NJ: Prentice Hall,1996.

LYNC93 Lynch,M.*Microprogrammed State Machine Design.* Boca Raton, FL:CRC Press,1993

PARK89 Parker, A., *and Hamblen,*J. *An Introduction to microprogramming with Exercises Designed for the Texas Instruments SN74ACT8800 Software Development Board.* Dallas,TX: Texas Instruments,1989.

SEGE91 Segee, B.,and Field,J. *Microprogramming and Computer Architecture.* New York: Wiley,1991.

TI90 Texas Instruments Inc. *SN74ACT880 Family Data Manual.*SCSS006C,1990

BELL71a Bell, C., and Newell, *A Computer Structures: Readings* and Examples. New York: McGraw-Hill, 1971.

BELL78a Bell, C.; Mudge, J.; and McNamara, J. *Computer Enginee*ring: A DEC *View of Hardware Systems Design*. Bedford, MA: Digital Press, 1978.

BETK97 Betker, M.: Fernando, J.;and Whalen, S.”The History of the Microprocessor.” *Bell Labs Technical Journal*, Autumn 1997.

BLAA97 Blaauw, G., and Brooks, F. *Computer Architecture: Concepts and Evolution*. Reading, MA:Addison-Wesley,1997.

BOHR98 Bohr, M. “Silicon Trends and Limits for Advanced Microprocessors.” *Communications of The ACM*, March 1998.

BREV97 Brey, B *The Intel Microprocessors: 8086/8066, 80186/80188, 80286, 80386, 80486, Pentium, and Pentiumprocessor*.Upper Saddle river, NJ: Prentice Hall, 1997.

HUTC96 Hutchson, G., and Hutcheson,j.” Technology and Economics in The Semiconductor Industry.” *Scientific American*, January 1996.

IBM94 International Lbusiness Machines, Inc. *The Powerpc Architecture: A Specification for a New Family of Risc Processors*. San Francisco,CA: Morgan Kaufmann, 1994.

INTE98a Intel Corp. *Pentium Processors and*

Related Products. Aurora, CO, 1998.

INTE98b Intel Corp. *Pentium Pro and Pentium II Processors and Related Products*. Aurora, CO,1998.

SCHA97 Schaller, R." Moore's Law: Past, Present, and Future."*IEEE Spectrum*, June 1997.

SHAN98 Shanley, T. *Pentium Pro and Pentium II System Architecture*. Reading, MA: Addison-Wesley, 1998.

SHAN95 Shanley, T. *Powerpc System Architecture*. Reading, MA: Addison-Wesley, 1995.

SIEW82 Siewiorek, D.,;Bell, C.;and Newell, A. *Computer Structures: Principles and Examples*. Newyork: Mcgraw-Hill, 1982.

WEIS94 Weiss, S., and Smith, J. *Power and Powerpc.* San Francisco:morgan Kaufmann, 1994.

색 인

ㅅ

ㅈ

ㅊ

ㅋ

ㅌ

ㅍ

략 어

ALU	산수−론리연산장치
ASCII	정보교환용미국표준코드
ANSI	미국국가표준협회
BCD	2 진식 10 진부호
CD	고밀도디스크
CD-ROM	고밀도디스크-읽기전용기억기
CPU	중앙처리장치
CISC	복합명령모임콤퓨터
DRAM	동적자유기억기
DMA	직접기억기접근
EPIC	명시적병렬명령계산
EPROM	지우기가능프로그람읽기전용기억기
EEPROM	명시적인 지우기가능프로그람읽기전용기억기
HLL	고급언어
I/O	입력/출력
IAR	명령주소등록기
IC	집적회로
IEEE	국제전기전자기술자협회
ILP	명령준위병렬화
IR	명령등록기
LRU	가장최근리용
LSI	대규모집적회로
MAR	기억기주소등록기
MBR	기억기완충등록기
MESI	변경-독점-공유-무효
MMU	기억기관리단위
MSI	중규모집적회로
NUMA	비균일기억기접근
PC	프로그람계수기
PCI	주변장치호상접속
PROM	프로그람식읽기전용기억기
PSW	처리장치상태단어
PCB	독립디스크들의 여유배렬
RALU	등록기/산수−론리연산장치
RAM	자유기억기
RISC	축소명령모임콤퓨터
ROM	읽기전용기억기
SCSI	작은 콤퓨터체계대면부
SMP	대칭다중처리장치
SRAM	정적자유기억기
SSI	소규모집적회로
VLSI	초대규모집적회로
VLWI	매우 긴 명령단어
WORM	쓰기-한번 읽기-여러번